Bachelor Thesis

Proof Systems for Regular Expressions

Manuel Schweigler

12108342

Supervised by
Associate Prof. Dipl.-Ing. Dr.techn. Stefan Hetzl

TU Wien
March 5, 2025

Abstract

We are interested in proving equations between regular expressions by applying substitutions defined by a set of axioms. The key insight of this thesis is that there exists no finite axiom system that can be used to prove all valid equations. To show this, we assume there exists such a finite proof system. By generalizing this system to a certain axiomatization, we can construct an equation that cannot be derived by this system.

Furthermore, we introduce two alternative proof systems that implement a finite axiomatization by introducing special transformations. Lastly, we analyze the complexity of proving the equality of regular expressions.

Contents

1 Introduction

In this thesis we analyze axiom systems which consist of equations between regular expressions. These axioms can be combined to generate new equations. In the first chapter, we derive some identities that are useful for describing valid equations [6], i.e. equations where both sides correspond to the same language. This begs the question whether there exists a system of finitely many axioms that can be used to derive any valid equation. Chapter 2 answers this question negatively [8]. In Chapter 3, we present two alternative proof systems [5, 7] that implement a finite axiomatization by introducing special transformations. Lastly, Chapter 4 analyzes the complexity of proving the equality of regular expressions [1, 4].

2 Identities of Regular Expressions

The proofs in this chapter are taken from [6].

Definition 2.1. Let A be an alphabet. A *regular expression* is a string of symbols and letters in A. Each regular expression X corresponds to a regular language in A^*, which we notate as $|X|$. The set of all regular expressions can be defined recursively:

1. Each letter $a \in A$ is a regular expression and describes the singleton $\{a\}$.
2. The symbol 0 matches the empty set $\emptyset$ and 1 corresponds to the set $\{\varepsilon\}$, where ε is the empty word in A^*.
3. If X and Y are regular expressions, then so are $X + Y$ and $X \cdot Y$. We define $|X+Y| \coloneqq |X| \cup |Y|$ and $|X \cdot Y| \coloneqq \{xy \mid x \in |X|, y \in |Y|\}$, i.e. the concatenation of all words in $|X|$ and $|Y|$. The dot in $X \cdot Y$ can be omitted.
4. If X is a regular expression, then so is X^*. The unary operation * is called the *Kleene-star*. Additionally, X^* corresponds to the set $\{\varepsilon\} \cup \bigcup_{n=1}^{\infty} X^n$, where $X^1 \coloneqq X$ and $X^n \coloneqq XX^{n-1}$.

An equation $X = Y$ of regular expressions is *valid* if $|X| = |Y|$. Our key goal is to construct a proof for a given valid equation $X = Y$. Hence, we first need to define a set $\mathcal{D}$ of axioms, i.e. $\mathcal{D}$ consists of some valid equations that can be combined in order to derive $X = Y$. Each step of the proof substitutes X for an equal expression by applying an axiom from $\mathcal{D}$, until we eventually reach an expression that is identical to Y.

Example 2.2. We want to prove the equation

$$1 + (\alpha + \beta)\alpha^* = (1 + \beta)\alpha^*$$

using the following axioms:

1. $(\alpha + \beta)\gamma = \alpha\gamma + \beta\gamma$
2. $1 + \alpha\alpha^* = \alpha^*$
3. $\alpha = 1 \cdot \alpha$

The proof consists of transforming the left side until it matches the right side.

$$\begin{aligned} 1 + (\alpha + \beta)\alpha^* &=^{1.} \\ 1 + \alpha\alpha^* + \beta\alpha^* &=^{2.} \\ \alpha^* + \beta\alpha^* &=^{3.} \\ 1 \cdot \alpha^* + \beta\alpha^* &=^{1.} \\ (1 + \beta)\alpha^* \end{aligned}$$

Different choices of $\mathcal{D}$ allow us to prove different sets of equations. This begs the question whether there exists a finite set of axioms that can be used to prove any valid equation. Chapter 3 addresses this question and proves that this is not the case. In the remainder of this chapter, we derive various identities that appear in later proofs.

2.1 Classical Identities

Regular expressions that do not contain the Kleene star operator satisfy all the properties of a semiring:

$$\alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma \tag{A+}$$
$$\alpha(\beta\gamma) = (\alpha\beta)\gamma \tag{A$\cdot$}$$
$$\alpha + \beta = \beta + \alpha \tag{C+}$$
$$\alpha + 0 = 0 + \alpha = \alpha \tag{U1}$$
$$\alpha \cdot 1 = 1 \cdot \alpha = \alpha \tag{U2}$$
$$\alpha \cdot 0 = 0 \cdot \alpha = 0 \tag{U3}$$
$$\alpha(\beta + \gamma) = \alpha\beta + \alpha\gamma \tag{DL}$$
$$(\alpha + \beta)\gamma = \alpha\gamma + \beta\gamma \tag{DR}$$

Furthermore, the operator $+$, which corresponds to the union of regular sets, satisfies the idempotent identity

$$\alpha + \alpha = \alpha. \tag{I}$$

These axioms suffice to prove equations that only use unions and concatenations. Note that such regular expressions always describe finite languages.

The following identities can be derived directly from Definition 2.1:

$$\alpha^* = 1 + \alpha\alpha^*$$
$$\alpha^* = 1 + \alpha^*\alpha$$
$$(\alpha^*)^* = \alpha^*$$
$$0^* = 1$$

In order to prove the remaining identities, we need to establish *Arden's Lemma*, which can be used to show that an equation is valid.

Lemma 2.3 (Arden)**.** Let K and L be two sets of words over the alphabet A^* such that K does not contain the word ε. Then $X = K^*L$ is the unique solution of the equation

$$X = KX + L \tag{1}$$

Proof. Firstly, we verify that K^*L is a valid solution by using the identities defined so far:

$$K^*L = (1 + KK^*)L = (KK^* + 1)L = K(K^*L) + L$$

In order to show uniqueness, let P be an arbitrary solution. From $P = KP + L$, we infer both $L \subseteq P$ and $KP \subseteq P$. By combining these two, we also see that $KL \subseteq P$. This principle can be continued inductively to prove $K^*L \subseteq P$. For the other direction, we verify $P \subseteq K^*L$ by contradictorily assuming that $P \setminus K^*L$ is not empty. Let w be a word in $P \setminus K^*L$ of minimal length. Since $K^*L \subseteq L$, w is not contained in L and thus $w \in KP$. Hence, there exist $k \in K$ and $p \in P$ such that $w = kp$. Furthermore, K does per definition not contain the empty word, so $k \neq \varepsilon$. Finally, p is strictly shorter than w and thus lies in K^*L, which implies that w is also in K^*L. □

2.2 Aperiodic Identities

Theorem 2.4. The following equations are valid:

$$(\alpha+\beta)^* = \alpha^*(\beta\alpha^*)^* \tag{S_l}$$
$$(\alpha+\beta)^* = (\alpha^*\beta)^*\alpha^* \tag{S_r}$$
$$(\alpha\beta)^* = 1+\alpha(\beta\alpha)^*\beta$$

Proof. According to Arden's Lemma, the equation

$$X = (\alpha+\beta)X + 1$$

has the unique solution $(\alpha+\beta)^*$. Thus, the identity (S_l) holds if and only if $\alpha^*(\beta\alpha^*)^*$ solves this equation:

$$\begin{aligned}(\alpha+\beta)(\alpha^*(\beta\alpha^*)^*)+1 &= \alpha\alpha^*(\beta\alpha^*)^* + \beta\alpha^*(\beta\alpha^*)^* + 1\\ &= \alpha\alpha^*(\beta\alpha^*)^* + (\beta\alpha^*)^* = (\alpha\alpha^*+1)(\beta\alpha^*)^* = \alpha^*(\beta\alpha^*)^*\end{aligned}$$

Note that the proof for $S_{\mathbf{r}}$ is analogous. The third identity is the unique solution of the equation

$$X = (\alpha\beta)X + 1$$

and can be proven in a similar manner. □

2.3 Cyclic Identities

The cyclic identities are a collection of infinitely many equations. These identities play an important part in Chapter 3, as axiom systems that contain the identities (Z_k) for all $k \geq 1$ are consequently infinite in size.

Theorem 2.5. The equation

$$\alpha^* = (\alpha^k)^*\alpha^{<k}, \tag{Z_k}$$

where $k \geq 1$ and $\alpha^{<k} \coloneqq (1+\alpha+\cdots+\alpha^{k-1})$, is valid.

Proof. As in the proof for the aperiodic identities, we show that

$$X = \alpha X + 1.$$

is solved by $(\alpha^k)^*\alpha^{<k}$ for each $k \geq 1$.

$$\begin{aligned}\alpha(\alpha^k)^*\alpha^{<k}+1 &= (\alpha^k)^*\alpha\alpha^{<(k-1)} + (\alpha^k)^*\alpha^k + 1\\ &= (\alpha^k)^*\alpha\alpha^{<(k-1)} + (\alpha^k)^*\\ &= (\alpha^k)^*\alpha^{<k}\end{aligned}$$

□

3 Equational Axiomatization

In the previous chapter, we analyzed some valid identities of regular expressions. This chapter proves that there exists no finite set of axioms that could be used to derive all valid equations. We give a slightly more detailed version of the proof given by Arto Salomaa in [8].

3.1 Preliminaries

We define regular expressions over the following alphabets:

$$I_r \coloneqq \{x_1, \ldots, x_r\}, \text{ for } r \geq 1$$
$$A_\omega \coloneqq \{\alpha, \beta, \gamma, \alpha_1, \beta_1, \gamma_1, \alpha_2, \ldots\}$$

The alphabet I_r is of size r, whereas A_ω has infinitely many elements. We interpret the letters in A_ω as placeholders for arbitrary regular expressions.

Furthermore, $\mathcal{V}_r$ denotes the set of all valid equations between regular expressions over the alphabet I_r. The set $\mathcal{S}_r$ consists of all equations $X = Y$ such that substituting every variable $a \in A_\omega$ with a regular expression $R(a)$ over I_r results in an equation $X_r = Y_r$ that is contained in $\mathcal{V}_r$.

We define $\mathcal{S}_\omega$ as the intersection of all $\mathcal{S}_r$ with $r \geq 1$. It is easy to see that the sets $\mathcal{S}_r$ follow a chain of inclusions:

$$\mathcal{S}_1 \supseteq \mathcal{S}_2 \supseteq \cdots \supseteq \mathcal{S}_\omega$$

The equation $\alpha\beta = \beta\alpha$ belongs to $\mathcal{S}_1$, but not to $\mathcal{S}_2$. Thus, $\mathcal{S}_1 \supsetneq \mathcal{S}_2$.

Theorem 3.1. $\mathcal{S}_2 = \mathcal{S}_3 = \cdots = \mathcal{S}_\omega$

Proof. It suffices to show that $\mathcal{S}_2 \subseteq \mathcal{S}_\omega$. Let $X = Y$ be an equation in $\mathcal{S}_2$ and assume that $X = Y \notin \mathcal{S}_\omega$. Hence, there must exist $r \geq 3$ such that $X = Y \notin \mathcal{S}_r$. Substituting the variables in $X = Y$ with certain regular expressions in I_r thus produces an equation $X_r = Y_r$ which is not valid. Therefore, without loss of generality, there must exist a word w such that $w \in |X_r|$ and $w \notin |Y_r|$.

We introduce a function $f : |I_r^*| \to |I_2^*|$, which maps each letter in a word to a block of r letters according to the following rules:

$$\begin{aligned}
f(x_1) &= x_1 x_2^{r-1} \\
f(x_2) &= x_2 x_1 x_2^{r-2} \\
&\vdots \\
f(x_r) &= x_2^{r-1} x_1 \\
f(\varepsilon) &= \varepsilon \\
f(pq) &= f(p)f(q), \text{ for } p, q \in |I_r^*|
\end{aligned}$$

For a regular expression Z over I_r, we define Z_f as the regular expression obtained by replacing each letter x_i by $f(x_i)$. Since f is injective, we conclude that for every

word $u \in |I_r^*|$ it holds that $u \in |Z|$ if and only if $f(u) \in |Z_f|$. Thus, we infer that $f(w) \in |(X_r)_f|$ and $f(w) \notin |(Y_r)_f|$. This, however, implies that $(X_r)_f = (Y_r)_f$ is not a valid equation. Since this equation can be generated from $X = Y$ using a substitution with regular expressions over I_2, $X = Y$ is not $\mathcal{S}_2$, which contradicts our initial assumption. □

Definition 3.2. Let X and Y be regular expressions over the same alphabet. We call X a *well-formed part* of Y if and only if one of the following conditions is met:

- $Y = X$ or $Y = X^*$
- There exists a regular expression Z such that $Y = (X + Z)$, $Y = (Z + X)$, $Y = XZ$ or $Y = ZX$.
- There exists a regular expression Z such that X is a well-formed part of Z and Z is a well-formed part of Y.

Definition 3.3. Let $\mathcal{D}$ be a set of equations and U be a well-formed part of a regular expression $X[U]$. Replacing U in X with an expression V results in the new term $X[V]$. If $X[U] = Y \in \mathcal{D}$ and $U = V \in \mathcal{D}$, then $X[V] = Y$ is called a *replacement*. If $\mathcal{D}$ is defined over the alphabet A_ω, each letter that appears in $X = Y$ can be substituted by some regular expressions. The resulting equation $X' = Y'$ is called a *substitution*.

Definition 3.4. Let $\mathcal{D}$ be a set of equations. An equation $X = Y$ is *generated* by $\mathcal{D}$, written as $\vdash_{\mathcal{D}} X = Y$, if $X = Y$ can be obtained from $\mathcal{D}$ by a finite number of replacements and substitutions. An equation $X = Y$ is *transformable* by $\mathcal{D}$ into $X' = Y'$ if $\mathcal{D}$ generates both $X = X'$ and $Y = Y'$. A subset $\mathcal{B} \subseteq \mathcal{D}$ is a *basis* of $\mathcal{D}$ if each equation in $\mathcal{D}$ can be generated by $\mathcal{B}$.

Definition 3.5. Let $r \geq 1$. For a word w over an alphabet A, we define the *Parikh vector* $p(w)$ as

$$\begin{aligned} p(w) : A &\to \mathbb{N}, \\ \alpha &\mapsto |w|_\alpha \end{aligned}$$

where $|w|_\alpha$ is the number of occurrences of α in w. Furthermore, for a regular expression X, we define

$$|X|_c \coloneqq \{w \in A^* \mid \exists v \in |X| \colon p(w) = p(v)\}.$$

An equation $X = Y$ is *c-valid* if $|X|_c = |Y|_c$.

Definition 3.6. For $r \geq 1$, $\mathcal{C}_r$ denotes the set of all equations $X = Y$ of regular expressions over A_ω that are c-valid when the letters appearing in the equation are substituted by regular expressions over I_r. We define $\mathcal{C}_\omega$ as the intersection of $\mathcal{C}_r$ for all $r \geq 1$.

Similarly to the set $\mathcal{S}_r$, these sets form a chain of inclusions:

$$\mathcal{C}_1 \supseteq \mathcal{C}_2 \supseteq \cdots \supseteq \mathcal{C}_\omega$$

Furthermore, it can be easily inferred that $\mathcal{S}_r \subseteq \mathcal{C}_r$ for all $r \geq 1$ and especially

$$\mathcal{S}_\omega \subseteq \mathcal{C}_\omega \tag{2}$$

In order to prove that the set $\mathcal{V}_r$ cannot be finitely axiomatized, we use $\mathcal{S}_\omega \subseteq \mathcal{C}_\omega$ to show that each possible basis $\mathcal{B}$ of $\mathcal{V}_r$ would also be a basis of $\mathcal{C}_\omega$. For that, we define a basis $\mathcal{T}$ of $\mathcal{C}_\omega$, which we will use to prove that $\mathcal{B}$ is not a basis of $\mathcal{V}_r$.

3.2 Basis $\mathcal{T}$

This subchapter proves that the set $\mathcal{T}$, as defined below, is a basis of the set $\mathcal{C}_\omega$.

Definition 3.7. The set $\mathcal{T}$ consists of the following equations:

$$\alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma \tag{A+}$$
$$\alpha(\beta\gamma) = (\alpha\beta)\gamma \tag{A$\cdot$}$$
$$\alpha + \beta = \beta + \alpha \tag{C+}$$
$$\alpha(\beta + \gamma) = \alpha\beta + \alpha\gamma \tag{DL}$$

$$\alpha + \alpha = \alpha \tag{I}$$
$$\alpha + 0 = \alpha \tag{U1}$$
$$\alpha \cdot 1 = \alpha \tag{U2}$$
$$\alpha \cdot 0 = 0 \tag{U3}$$

$$\alpha^* = (1 + \alpha)^* \tag{3}$$
$$\alpha^* = (\alpha^k)^*(1 + \alpha + \cdots + \alpha^{k-1}), \text{ for } k \geq 1 \tag{Z$_k$}$$

$$\alpha\beta = \beta\alpha \tag{C$\cdot$}$$
$$(\alpha\beta^*)^* = 1 + \alpha\alpha^*\beta^* \tag{4}$$
$$(\alpha + \beta)^* = \alpha^*\beta^* \tag{5}$$
$$(\alpha + \beta)^* = (\alpha\beta)^*(\alpha^* + \beta^*) \tag{6}$$

Lemma 3.8. Let $n \geq 2$ and $I = \{1, \ldots, n\}$. Then $\mathcal{T}$ generates the equation

$$\left(\sum_{i \in I} \alpha_i\right)^* = (\alpha_1 \cdots \alpha_n)^* \sum_{j \in I} \left(\sum_{i \in I \setminus \{j\}} \alpha_i\right)^*$$

Proof. We use an induction over n. For $n = 2$, the assertion follows from (6). If $n \geq 3$, the equation can be generated as follows:

$$\begin{aligned}
\left(\sum_{i\in I} \alpha_i + \alpha_{n+1}\right)^* &= (\sum_{i\in I} \alpha_i)^* \alpha_{n+1}^* \\
&= (\alpha_1 \cdots \alpha_n)^* \sum_{j\in I} \left(\sum_{i\in I\setminus\{j\}} \alpha_i\right)^* \alpha_{n+1}^* \\
&= (\alpha_1 \cdots \alpha_n + \alpha_{n+1})^* \sum_{j\in I} \left(\sum_{i\in I\setminus\{j\}} \alpha_i\right)^* \\
&= (\alpha_1 \cdots \alpha_n \alpha_{n+1})^* \left((\alpha_1 \cdots \alpha_n)^* + \alpha_{n+1}^*\right) \sum_{j\in I} \left(\sum_{i\in I\setminus\{j\}} \alpha_i\right)^* \\
&= (\alpha_1 \cdots \alpha_{n+1})^* \left(\left(\sum_{i\in I} \alpha_i\right)^* + \sum_{j\in I} \left(\sum_{i\in I\setminus\{j\}} \alpha_i + \alpha_{n+1}\right)^*\right) \\
&= (\alpha_1 \cdots \alpha_{n+1})^* \sum_{j\in I\cup\{n+1\}} \left(\sum_{i\in I\cup\{n+1\}\setminus\{j\}} \alpha_i\right)^*
\end{aligned}$$

□

Definition 3.9. Let $n \geq 0$. Furthermore, let $P_1, \ldots, P_{n+1}$ be words over A_ω. If $P_1, \ldots, P_n$ are not empty, then

$$(P_1 + \ldots + P_n)^* P_{n+1}$$

is called a *normal product*. For $n = 0$, this expression is reduced to P_1. A normal product is *linearly independent* if the Parikh vectors $p(P_1), \ldots, p(P_n)$ are linearly independent.

Lemma 3.10. Let P be a normal product. There exists a sum S of linearly independent normal products such that $P = S$ is generated by $\mathcal{T}$.

Proof. Let $P = (P_1 + \ldots + P_n)^* P_{n+1}$. If P is linearly independent, we can choose $P = S$. Otherwise, P is linearly dependent and we can infer without loss of generality that

$$P_1^{i_1} \cdots P_v^{i_v} = P_{v+1}^{i_{v+1}} \cdots P_n^{i_n} \tag{7}$$

holds for some $v \leq n$, $i_1, \ldots, i_v > 0$ and $i_{v+1}, \ldots, i_n \geq 0$.

Combining Lemma 3.8 with (7) yields the following equation:

$$(P_1^{i_1} + \ldots + P_v^{i_v})^* = (P_{v+1}^{i_{v+1}} \cdots P_n^{i_n})^* \sum_{j\in\{1,\ldots,v\}} \Big(\sum_{k\in\{1,\ldots,v\}\setminus\{j\}} P_k^{i_k}\Big)^*$$

Furthermore, $\mathcal{T}$ generates the following equation using identities (3) and (5):

$$(P_{v+1}^{i_{v+1}} \cdots P_n^{i_n})^* (P_{v+1}^{i_{v+1}} + \ldots + P_n^{i_n})^* = (P_{v+1}^{i_{v+1}} + \ldots + P_n^{i_n})^*$$

Finally, we can transform our normal product P. Using $W_j = (P_j^0 + \ldots + P_j^{i_j - 1})$ and $W = W_1 \cdots W_n P_{n+1}$, we can rewrite P as

$$\begin{aligned}
(P_1 + \ldots + P_n)^* P_{n+1} &= P_1^* \cdots P_n^* P_{n+1} \\
&=^{(\mathrm{Z}_k)} (P_1^{i_1})^* W_1 \cdots (P_1^{i_n})^* W_n P_{n+1} \\
&= (P_1^{i_1} + \ldots + P_v^{i_v})^* (P_{v+1}^{i_{v+1}} + \ldots + P_n^{i_n})^* W \\
&= \sum_{j \in \{1,\ldots,v\}} \Big(\sum_{k \in \{1,\ldots,v\} \setminus \{j\}} P_k^{i_k} \Big)^* (P_{v+1}^{i_{v+1}} + \ldots + P_n^{i_n})^* W \\
&= \sum_{j \in \{1,\ldots,v\}} \Big(\sum_{k \in \{1,\ldots,n\} \setminus \{j\}} P_k^{i_k} \Big)^* W
\end{aligned}$$

The final term forms a sum of normal products. Each product has strictly less than n summands. Since a normal product of the form $P_1^* P_2$ is linearly independent, we can reapply this procedure until we are left with a linearly independent sum S. □

We define:

$$\alpha \leq \beta \iff \alpha + \beta = \beta$$

Lemma 3.11. Let P and Q be linearly independent normal products. If $|P|_c \subseteq |Q|_c$, then the equation $P \leq Q$ is derivable in $\mathcal{T}$.

Proof. Let $P = (P_1 + \ldots + P_n)^* P_{n+1}$ and $Q = (Q_1 + \ldots + Q_m)^* Q_{m+1}$. We derive $P \leq Q$ by extending P to a normal product R such that $|P|_c \subseteq |R|_c$ and show that $P \leq R$ is derivable in $\mathcal{T}$. Because the relation $\leq$ is transitive, we repeat this procedure for R and Q until we eventually reach the equation $Q \leq Q$.

Since $|P_{n+1}|_c \in |Q|_c$, we infer that

$$P_{n+1} = \left(Q_1^{i_1} \cdots Q_m^{i_m} \right) Q_{m+1}$$

for some $i_1, \ldots, i_m$. Furthermore, for every $k \leq n$ we can define indices $j_{k,1}, \ldots, j_{k,m}$ such that

$$P_k = Q_1^{j_{k,1}} \cdots Q_m^{j_{k,m}}.$$

Let

$$R \coloneqq (P_1 + \ldots + P_n + Q_1 + \ldots + Q_m)^* P_{n+1}.$$

Then

$$\begin{aligned}
P + R &= P + (Q_1 + \ldots + Q_m + P_1 + \ldots + P_n)^* P_{n+1} \\
&= P + (Q_1 + \ldots + Q_m)^* P = (1 + (Q_1 + \ldots + Q_m)^*) P \\
&= (Q_1 + \ldots + Q_m)^* P = R
\end{aligned}$$

and thus $P \leq R$.

Next, we remove the factors $P_1, \ldots, P_n$ from R. Choose $k \leq n$ and $l \leq m$ such that $P_k \neq \varepsilon$ and $j_{k,l} \neq 0$. Let

$$R' \coloneqq (P_1 + \ldots + P_k' + \ldots + P_n + Q_1 + \ldots + Q_m)^* P_{n+1}$$

such that $P_k = Q_l P'_k$. Then

$$\begin{aligned}
R + R' &= (P_k^* + (P'_k)^*)\,[(Q_1 + \ldots + Q_m)^*(\cdots + P_{k-1} + P_{k+1} + \ldots)^* P_{n+1}] \\
&= ((Q_l P'_k)^* + (P'_k)^*) Q_l^*\,[\ldots] \\
&= ((Q_l P'_k)^* Q_l^* + (P'_k)^* Q_l^*)\,[\ldots] \\
&= \left((Q_l P'_k)^* Q_l^* + (Q_l + P'_k)^*\right)[\ldots] \\
&=^{(6)} \left((Q_l P'_k)^* Q_l^* + (Q_l P'_k)^* (Q_l^* + P_k^*)\right)[\ldots] \\
&= \left((Q_l P'_k)^* (Q_l^* + P_k^*)\right)[\ldots] \\
&= (Q_l + P_k)^*\,[\ldots] \\
&= (P'_k)^* Q_l^*\,[\ldots] \\
&= R'
\end{aligned}$$

We apply this procedure repeatedly until we get a product R'' of the form

$$R'' \coloneqq (Q_1 + \ldots + Q_m)^* P_{n+1}.$$

In order to correct the last factor, we rewrite it using the form $P_{n+1} = \left(Q_1^{i_1} \cdots Q_m^{i_m}\right) Q_{m+1}$. Choose $k \leq n$ such that $i_k \neq 0$ and let

$$R''' \coloneqq (Q_1 + \ldots + Q_m)^* P'''_{m+1},$$

where $P'''_{m+1} Q_k = P_{n+1}$. Hence,

$$\begin{aligned}
R'' + R''' &= (Q_1 + \ldots + Q_m)^* (P'''_{m+1} Q_k + P'''_{m+1}) \\
&= (Q_1 + \ldots + Q_m)^* (Q_k + 1) P'''_{m+1} \\
&= (Q_1 + \ldots + Q_m)^* (Q_k 1)^* (Q_k + 1) P'''_{m+1} \\
&=^{(6)} (Q_1 + \ldots + Q_m)^* (Q_k + 1)^* P'''_{m+1} \\
&= (Q_1 + \ldots + Q_m)^* Q_k^* P'''_{m+1} \\
&= R'''.
\end{aligned}$$

After repeated application, we can show $\vdash_{\mathcal{T}} R''' \leq Q$. Since $\leq$ is transitive, we have thus proven that $\vdash_{\mathcal{T}} P \leq Q$. □

Definition 3.12. For linearly independent normal products $P = (P_1 + \ldots + P_n)^* P_{n+1}$ and $Q = (Q_1 + \ldots + Q_m)^* Q_{m+1}$, we say that Q *spans over* P if for each $j \leq n$, there exists $t_j \geq 1$ such that

$$|(P_j)^{t_j}|_c \in |(Q_1 + \ldots + Q_m)^*|_c. \tag{8}$$

Lemma 3.13. Let P and $Q^{(1)}, \ldots, Q^{(k)}$ be linearly independent normal products. If $|P|_c \subseteq |Q^{(1)} + \ldots + Q^{(k)}|_c$, then the equation $P \leq Q^{(1)} + \ldots + Q^{(k)}$ is derivable in $\mathcal{T}$.

Proof. Let $P = (P_1 + \ldots + P_n)^* P_{n+1}$ and $\mathcal{Q} \coloneqq (Q^{(1)}, \ldots, Q^{(k)})$. We prove this statement inductively over n. For $n = 0$, P consists of a single word. Hence, there exists $i \leq k$ such that $|P|_c \subseteq |Q^{(i)}|_c$ and by Lemma 3.11 $\mathcal{T}$ generates $P \leq Q^{(i)} \leq Q^{(1)} + \ldots + Q^{(k)}$.

Next, let $n \geq 1$. Let $\mathcal{Q}'$ be the set of all products in $\mathcal{Q}$ that span over P. We claim that there exists a string $S \in |(P_1 + \ldots + P_n)^*|$ such that $|PS|_c \subseteq |\sum_{Q \in \mathcal{Q}'} Q|_c$. To prove this, suppose there exist a set $\mathcal{Q}_S \subseteq \mathcal{Q}$ and $i \leq k$ such that $Q^{(i)}$ does not span over P, $Q^{(i)} \in \mathcal{Q}_S$, $\mathcal{Q}_S$ covers PS and $\mathcal{Q}_S \setminus Q^{(i)}$ does not cover PS. Additionally, choose $j \leq n$ such that $(P_j)^{t_j} \notin |(Q_1^{(i)} + \ldots + Q_m^{(i)})^*|_c$ for all $t_j \geq 1$. From this we conclude $|Q^{(i)}|_c \cap |Q^{(i)} P_j|_c = \emptyset$. Moreover, this implies $|Q^{(i)}|_c \cap |PSP_j|_c = \emptyset$. Thus, we can define $S' = SP_j$ and $\mathcal{Q}_{S'} = \mathcal{Q}_S \setminus \{Q^{(i)}\}$, resulting in $|PS'|_c \subseteq |\sum_{Q \in \mathcal{Q}_{S'}} Q|_c$.

For each $i \leq k$ that satisfies $Q^{(i)} \in \mathcal{Q}'$, we define the linearly independent normal product

$$R^{(i)} \coloneqq (P_1^{t_1^{(i)}} + \ldots + P_n^{t_n^{(i)}})^* P_{n+1} S (P_1^{u_1^{(i)}} \cdots P_n^{u_n^{(i)}}),$$

where $t_j^{(i)}$ is obtained by applying Definition 3.12 to P and $Q^{(i)}$, and $0 \leq u_j^{(i)} \leq t_j^{(i)}$. This expression satisfies $|R^{(i)}|_c = |Q^{(i)}|_c \cap |PS|_c$. Furthermore, for each $j \leq n$, define τ_j as the least common multiple of $\{t_j^{(i)} \mid i \leq k, Q^{(i)} \in \mathcal{Q}'\}$. Using the notation $X^{<n} \coloneqq 1 + X + \ldots + X^{n-1}$, we can rewrite $R^{(i)}$ as

$$\begin{aligned} R^{(i)} &= \prod_{j=1}^{n} \left(P_j^{t_j^{(i)}} \right)^* P_j^{u_j^{(i)}} P_{n+1} S \\ &=^{(\mathrm{Z})} \prod_{j=1}^{n} \left(\left(P_j^{t_j^{(i)}} \right)^{<\left(\frac{\tau_j}{t_j^{(i)}} \right)} \right) P_j^{u_j^{(i)}} \left(P_j^{\tau_j} \right)^* P_{n+1} S \end{aligned}$$

Thus, define

$$T(v_1, \ldots, v_n) = \prod_{j=1}^{n} P_j^{v_j} \left(P_j^{\tau_j} \right)^* P_{n+1} S.$$

Each product $R^{(i)}$ can be transformed into a sum of terms $T(\ldots)$. For all $\mathbf{v} = (v_1, \ldots, v_n)$ such that $0 \leq v_j < \tau_j$ and $Q \in \mathcal{Q}'$, this definition satisfies $|T(v_1, \ldots, v_n)|_c \subseteq |Q|_c$. By Lemma 3.11 follows $\vdash_{\mathcal{T}} T(v_1, \ldots, v_n) \leq (Q^{(1)} + \ldots + Q^{(k)})$. Note that PS is equal to the sum of all possible expressions $T(\ldots)$. Hence, we can split PS into terms $T(\ldots)$ using the identity (Z), resulting in

$$\vdash_{\mathcal{T}} PS = \sum_{\mathbf{v} \leq \tau} T(v_1, \ldots, v_n) \leq (Q^{(1)} + \ldots + Q^{(k)}).$$

In order to generate the words in $|P|_c \setminus |PS|_c$, we use our induction hypothesis. For $S = P_i S_i$, the set $|PS_i|_c \setminus |PS|_c$ can be described by the linearly independent normal product

$$U_i = \left(\sum_{j=1, j \neq i}^{n} P_j \right)^* P_{n+1} S_i,$$

which uses one less element than P. Thus, by induction $U_i \leq Q^{(1)} + \ldots + Q^{(k)}$. Furthermore, since $U_i \leq PS_i$, we get

$$\begin{aligned} PS_i &= \left(\sum_{j=1}^{n} P_j\right)^* P_{n+1}S_i \\ &= \left(\sum_{j=1,j\neq i}^{n} P_j\right)^* P_{n+1}S_iP_i^* \\ &=^{(4)} \left(\sum_{j=1,j\neq i}^{n} P_j\right)^* P_{n+1}S_i(1 + P_iP_i^*) \\ &= PS + U_i. \end{aligned}$$

Applying this method repeatedly produces

$$\vdash_{\mathcal{T}} P \leq PS + \mathcal{U} \leq Q^{(1)} + \ldots + Q^{(k)},$$

where $\mathcal{U}$ is the sum of the products U_i that are used. □

Definition 3.14. For a regular expression, its *star height* is a non-negative integer that is defined recursively according to the following rules:

1. An expression that consists of a single letter or is the empty term ε has star height 0.
2. If X and Y are regular expressions with star heights i and j respectively, the terms $X + Y$ and XY both have star height $\max(i, j)$.
3. If X has star height i, then X^* has star height $i + 1$.

Lemma 3.15. Let $X = Y$ be an equation in $\mathcal{C}_\omega$. Then there exist regular expressions X_1 and Y_1 that consist of finite sums of normal products and satisfy

$$\begin{aligned} \vdash_{\mathcal{T}} X &= X_1 \\ \vdash_{\mathcal{T}} Y &= Y_1. \end{aligned}$$

Hence, $X = Y$ can be transformed into the form $X_1 = Y_1$.

Proof. We want to transform both sides of this equation to a regular expression with star height 1, which is always possible using the axioms in $\mathcal{T}$. To prove this, it suffices to show that any regular expression Z^* of star height 2 can be transformed into an equivalent form Z' with star height 1. By applying this method recursively, it can be used on expressions of any star height, especially on X and Y. Note that an expression of star height 1 can be easily converted using (DL) and (C·) into a finite sum of normal products.

Let Z^* be a regular expression of star height 2. By expanding Z, we can convert Z^* to the form

$$Z^* = \left(\sum_{i=1}^{m} Z_{i,1} \cdots Z_{i,n_i}\right)^* =^{(5)} \prod_{i=1}^{m} (Z_{i,1} \cdots Z_{i,n_i})^*.$$

Each element $Z_{i,j}$ is either a word or an expression of the form $W_{i,j}^*$.

Now, we analyze this expression for each $i \leq m$. If there exists no $j \leq n_i$ such that $X_{i,j}$ forms an expression with a Kleene star, the expression $(Z_{i,1} \cdots Z_{i,n_i})^*$ already has star height 1. Otherwise, $Z_{i,j}$ is of the form $W_{i,j}^*$ and we can apply the following transformation:

$$\begin{aligned}(Z_{i,1} \cdots W_{i,j}^* \cdots Z_{i,n_i})^* &= (Z_{i,1} \cdots Z_{i,j-1} Z_{i,j+1} \cdots Z_{i,n_i} W_{i,j}^*)^* \\ &=^{(4)} 1 + Z_{i,1} \cdots Z_{i,j-1} Z_{i,j+1} \cdots Z_{i,n_i} (Z_{i,1} \cdots Z_{i,j-1} Z_{i,j+1} \cdots Z_{i,n_i})^* W_{i,j}^*\end{aligned}$$

If there exists more than one possible index j, we can simply perform this method as many times as required. □

Definition 3.16. For a regular expression X, we define $\widetilde{\Lambda}(X)$ as the set of all linearly independent normal products P such that $|P|_c \subseteq |X|_c$. Furthermore, define the regular expression $\Lambda(X)$ as

$$\Lambda(X) \coloneqq \sum \{P \in \widetilde{\Lambda}(X) \mid \nexists Q \in \widetilde{\Lambda}(X) \colon |P|_c \subsetneq |Q|_c\}.$$

Theorem 3.17. The set $\mathcal{T}$ is a basis of $\mathcal{C}_\omega$.

Proof. Let $X = Y$ be an equation in $\mathcal{C}_\omega$. Using Lemma 3.15, this equation can be converted into a form $X_1 = Y_1$, where both sides are sums of finitely many normal products. Lemma 3.10 allows us to transform this equation further into a form $X_2 = Y_2$ such that all normal products are linearly independent.

Let $X_2 = Q^{(1)} + \ldots + Q^{(k)}$. For each product P in $\Lambda(X_2)$, $|P|_c$ is a subset of $|Q^{(1)} + \ldots + Q^{(k)}|_c$. Hence, by Lemma 3.13 we can generate the equation

$$X_2 = X_2 + \Lambda(X_2),$$

and, by symmetry,

$$Y_2 = Y_2 + \Lambda(Y_2).$$

Additionally, Lemma 3.11 proves

$$\begin{aligned}\vdash_{\mathcal{T}} X_2 + \Lambda(X_2) = \Lambda(X_2), \\ \vdash_{\mathcal{T}} Y_2 + \Lambda(Y_2) = \Lambda(Y_2).\end{aligned}$$

Lastly, note that the expressions $\Lambda(X_2)$ and $\Lambda(Y_2)$ are identical, since $|X_2|_c = |Y_2|_c$. □

3.3 Finite Axiomatizability

Definition 3.18. Let $p \geq 2$. A regular expression X over the alphabet $\{\alpha\}$ has the *p-property* if whenever Y^* is a well-formed part of X, then every word in $|Y|$ is of length divisible by p. We call an equation $X = Y$ *p-preserving* if either both X and Y or neither X nor Y satisfy the p-property.

For a prime p, the set $\mathcal{T}_p$ is constructed by removing the equation (Z_k) from $\mathcal{T}$ for all $k \geq p$.

Lemma 3.19. The equation

$$\alpha^* = (\alpha^p)^*(1 + \alpha + \ldots + \alpha^{p-1}) \qquad (\mathrm{Z}_p)$$

is not generated by the set $\mathcal{T}_p$.

Proof. We show that $\mathcal{T}_p$ only generates p-preserving equations. Firstly, this property can be easily verified for each substitution instance of any identity in $\mathcal{T}_p$. Additionally, let $X[U] = Y$ and $U = V$ be p-preserving equations generated by $\mathcal{T}_p$ such that U is a well-formed part of X. Then $X[V] = Y$ is also p-preserving. Since the equation Z_p is not p-preserving, it thus cannot be generated by $\mathcal{T}_p$. □

Theorem 3.20. The set $\mathcal{V}_2$ is not finitely generated. Hence, none of the sets $\mathcal{V}_r$ for $r \geq 2$ are finitely generated.

Proof. Assume $\mathcal{V}_2$ has a finite basis $\mathcal{B} \subseteq \mathcal{S}_2$. From $\mathcal{S}_2 = \mathcal{S}_\omega$ and $\mathcal{S}_\omega \subseteq \mathcal{C}_\omega$, we infer $\mathcal{B} \subseteq \mathcal{C}_\omega$. By Theorem 3.17, the set $\mathcal{T}$ generates $\mathcal{C}_\omega$. Since $\mathcal{B}$ is finite, there must exist a finite subset $\mathcal{T}' \subseteq \mathcal{T}$ that generates every equation in $\mathcal{B}$. Thus, the set $\mathcal{T}'$ forms a finite basis of $\mathcal{V}_2$.

Additionally, there exists a prime p such that $\mathcal{T}' \subseteq \mathcal{T}_p$. However, the axiom (Z_p) lies in $\mathcal{V}_2$ but cannot be generated by $\mathcal{T}_p \supseteq \mathcal{T}'$, as proven in Lemma 3.19. Thus, there exists no finite basis for $\mathcal{V}_2$.

Finally, the second part of the theorem follows from Theorem 3.1 and the fact that $\mathcal{V}_2 \subseteq \mathcal{V}_r$ for all $r \geq 2$. □

Corollary 3.21. There exists no finite axiomatization for valid equations of regular expressions.

4 Alternative Proof Systems

Theorem 3.20 proves that there exists no finite set of axioms such that every correct equation of regular expressions can be proven using replacements and substitutions. However, there are systems that achieve a finite axiomatization by introducing special transformations. In this chapter, we introduce two such systems.

4.1 Axiom System F_1

The system F_1 was introduced by Arto Salomaa in [7]. It contains the following axioms:

$$\alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma \tag{A+}$$
$$\alpha(\beta\gamma) = (\alpha\beta)\gamma \tag{A$\cdot$}$$
$$\alpha + \beta = \beta + \alpha \tag{C+}$$
$$\alpha(\beta + \gamma) = \alpha\beta + \alpha\gamma \tag{DL}$$
$$(\alpha + \beta)\gamma = \alpha\gamma + \beta\gamma \tag{DR}$$

$$\alpha + \alpha = \alpha \tag{I}$$
$$\alpha + 0 = \alpha \tag{U1}$$
$$\alpha \cdot 1 = \alpha \tag{U2}$$
$$\alpha \cdot 0 = 0 \tag{U3}$$

$$\alpha^* = (1 + \alpha)^* \tag{9}$$
$$\alpha^* = 1 + \alpha^*\alpha \tag{10}$$

We say that a regular expression X satisfies the *empty word property*, or e.w.p. for short, if $\varepsilon \in |X|$. This property can also be defined recursively. A regular expression α possesses e.w.p. if one of the following conditions is met:

- $\alpha = 1$
- $\alpha = \beta^*$ for some β
- $\alpha = \beta + \gamma$, where β e.w.p. $\vee$ γ e.w.p.
- $\alpha = \beta\gamma$, where β e.w.p. $\wedge$ γ e.w.p.

The system F_1 introduces a special transformation S, which yields the unique solution of the equation $\alpha = \alpha\beta + \gamma$ as stated in Lemma 2.3.

$$\frac{\neg(\beta \text{ e.w.p.}) \quad \alpha = \alpha\beta + \gamma}{\alpha = \gamma\beta^*} \tag{S}$$

Notably, this operation cannot be expressed using replacements and substitutions.

A *proof* in F_1 is a finite list of equations, where each step is either an axiom in F_1 or the application of a replacement, a substitution or the transformation S on one of the previous equations. An equation $X = Y$ is *derivable* in F_1, denoted by $\vdash_{F_1} X = Y$, if there exists a proof that ends in $X = Y$. The axiom system F_1 is *sound* if each derivable equation is valid, and *complete* if each valid equation is derivable.

Theorem 4.1. The axiom system F_1 is sound.

Proof. The axioms in F_1 are valid, as they are already proven in Chapter 2. Moreover, the validity of substitutions and replacements is obvious. Substituting α for $\gamma\beta^*$ in $\alpha = \alpha\beta + \gamma$ produces the equation $\gamma(\beta^*) = \gamma(\beta^*\beta + 1)$, which is valid because of identity (10). □

We write

$$\vdash_{F_1} (\alpha, \beta) = (\gamma, \delta)$$

to note that $\vdash_{F_1} \alpha = \gamma$ and $\vdash_{F_1} \beta = \delta$.

Lemma 4.2. If

$$\forall i \leq n\colon\ \vdash_{F_1} (\alpha_i, \beta_i) = \sum_{j=1}^{n} (\alpha_j, \beta_j)\gamma_{ij} + (\gamma_i, \gamma_i)$$

and none of the regular expressions γ_{ij} possesses the empty word property, then

$$\forall i \leq n\colon\ \alpha_i = \beta_i.$$

Proof. We use an induction over n. For $n = 1$, the expression has the form

$$\vdash_{F_1} (\alpha_1, \beta_1) = (\alpha_1, \beta_1)\gamma_{11} + (\gamma_1, \gamma_1).$$

Applying the transformation (S) yields

$$\vdash_{F_1} (\alpha_1, \beta_1) = (\gamma_1(\gamma_{11})^*, \gamma_1(\gamma_{11})^*)$$

and thus $\vdash_{F_1} \alpha_1 = \beta_1$.

For $n \geq 2$, the equation for the case $i = n$ can be transformed to

$$\begin{aligned}\vdash_{F_1} (\alpha_n, \beta_n) &= (\alpha_n, \beta_n)\gamma_{nn} + \left(\sum_{j=1}^{n-1} (\alpha_j, \beta_j)\gamma_{nj} + (\gamma_n, \gamma_n)\right) \\ &=^{(S)} \left(\sum_{j=1}^{n-1} (\alpha_j, \beta_j)\gamma_{nj} + (\gamma_n, \gamma_n)\right) (\gamma_{nn})^*.\end{aligned}$$

Substituting (α_n, β_n) in the original equation thus yields

$$\forall i \leq n-1\colon\ \vdash_{F_1} (\alpha_i, \beta_i) = \sum_{j=1}^{n-1} (\alpha_j, \beta_j)\,(\gamma_{ij} + \gamma_{nj}(\gamma_{nn})^*\gamma_{in}) + (\widetilde{\gamma}_i, \widetilde{\gamma}_i),$$

where $\widetilde{\gamma}_i \coloneqq \gamma_i + \gamma_n(\gamma_{nn})^*\gamma_{in}$. Since each coefficient does not possesses e.w.p., we can apply the induction case for $n - 1$. □

Definition 4.3. A regular expression α over the alphabet I_r is *equationally characterized* if there exist regular expressions $\alpha_1, \ldots, \alpha_n$ such that $\alpha = \alpha_1$ and

$$\forall i \leq n\colon\ \vdash_{F_1} \alpha_i = \sum_{j=1}^{r} \alpha_{ij}x_j + \delta(\alpha_i),$$

where $\delta(\alpha_i) \in \{0, 1\}$ and for each i and j, there exists $k \leq n$ such that $\alpha_{ij} = \alpha_k$.

Lemma 4.4. Let $\alpha = \beta$ be a valid equation. If

$$\vdash_{F_1} (\alpha, \beta) = \sum_{j=1}^{r} (\alpha_j, \beta_j) x_j + (\delta(\alpha), \delta(\beta)) \tag{11}$$

such that $\delta(\alpha), \delta(\beta) \in \{0, 1\}$, then $\delta(\alpha) = \delta(\beta)$ and the equation $\alpha_j = \beta_j$ is valid for all $j \leq r$.

Proof. By Theorem 4.1, equation (11) must be valid. As the terms $(\alpha_j, \beta_j)x_j$ for $j \leq r$ do not satisfy the e.w.p., $\delta(\alpha) = 1$ if and only if α possesses the empty word property. Additionally, since $\alpha = \beta$ is valid, $\delta(\alpha)$ and $\delta(\beta)$ must have the same value. Furthermore, the sets $|\alpha_j x_j|$ for $j \leq r$ are disjoint. Hence, the equations $\alpha_j = \beta_j$ must be valid. □

Lemma 4.5. Every regular expression is equationally characterized.

Proof. Firstly, each one-symbol expression is equationally characterized:

$$\vdash_{F_1} 0 = \sum_{j=1}^{r} 0x_j + 0$$

$$\vdash_{F_1} 1 = \sum_{j=1}^{r} 0x_j + 1$$

$$\forall i \leq r\colon \ \vdash_{F_1} x_i = 0x_1 + \ldots + 1x_i + \ldots + x_r + 0$$

Let α, β be regular expressions that are equationally characterized. We want to show that the terms $\alpha + \beta$, $\alpha\beta$ and α^* are also equationally characterized.

For $u \leq n$ and $v \leq m$, we define $\xi(u, v) \coloneqq \alpha_u + \beta_v$. Since

$$\begin{aligned} &\vdash_{F_1} 0 + 0 = 0 \\ &\vdash_{F_1} 0 + 1 = 1 + 0 = 1 \\ &\vdash_{F_1} 1 + 1 = 1, \end{aligned}$$

we can expand α_u and β_v in order to derive

$$\vdash_{F_1} \xi(u, v) = \sum_{j=1}^{r} (\alpha_{uj} + \beta_{vj}) x_j + \delta(u, v),$$

where $\delta(u, v) \in \{0, 1\}$. There exist $k \leq n$ and $l \leq m$ such that $\alpha_{uj} = \alpha_k$ and $\beta_{vj} = \beta_l$. Because $\alpha_{uj} + \beta_{vj} = \alpha_k + \beta_l = \xi(k, l)$ and $\xi(1, 1) = \alpha + \beta$, we know that $\alpha + \beta$ is equationally characterized.

Next, we define

$$\eta(u, (v_1, \ldots, v_h)) \coloneqq \alpha\beta_u + \alpha_{v_1} + \ldots + \alpha_{v_h},$$

where $u \leq m$, $h \geq 0$ and $1 \leq v_1 < v_2 < \cdots < v_h \leq n$. This expression can be transformed to

$$\begin{aligned}\vdash_{F_1} \eta(u,(v_1,\ldots,v_h)) &= \sum_{j=1}^{r} (\alpha\beta_{uj} + \alpha_{v_1 j} + \ldots + \alpha_{v_h j})\, x_j + \alpha\delta(\beta_u) + \delta(u,(v_1,\ldots,v_h)) \\ &= \sum_{j=1}^{r} (\alpha\beta_{uj} + \alpha_{1j}\delta(\beta_u) + \alpha_{v_1 j} + \ldots + \alpha_{v_h j})\, x_j + \delta(u,(v_1,\ldots,v_h)),\end{aligned}$$

where $\delta(u,(v_1,\ldots,v_h)) \in \{0,1\}$. Since the coefficients in the last equation are outputs of η and $\eta(1,()) = \alpha\beta$, we have shown that $\alpha\beta$ is equationally characterized.

Lastly, denote

$$\begin{aligned}\zeta() &\coloneqq \alpha^*, \\ \zeta(v_1,\ldots,v_n) &\coloneqq \alpha^*(\alpha_{v_1} + \ldots + \alpha_{v_h}).\end{aligned}$$

for $h \geq 1$ and $1 \leq v_1 < v_2 < \cdots < v_h \leq n$. As $\vdash_{F_1} (\alpha + 1)^* = \alpha^*$, we conclude

$$\begin{aligned}\vdash_{F_1} \alpha &= \sum_{j=1}^{r} \alpha_{1j} x_j + \delta(\alpha_1) \\ \Longrightarrow \vdash_{F_1} \alpha^* &= \left(\sum_{j=1}^{r} \alpha_{1j} x_j\right)^*.\end{aligned}$$

Applying $\alpha^* = 1 + \alpha^*\alpha$ yields

$$\vdash_{F_1} \zeta() = 1 + \left(\sum_{j=1}^{r} \alpha_{1j} x_j\right)^* \sum_{j=1}^{r} \alpha_{1j} x_j = \sum_{j=1}^{r} \alpha^*\alpha_{1j} x_j + 1.$$

Furthermore, we get

$$\vdash_{F_1} \zeta(v_1,\ldots,v_n) = \sum_{j=1}^{r} \alpha^*(\alpha_{1j}\delta(v_1,\ldots,v_n) + a_{v_1 j} + \ldots + a_{v_h j})x_j + \delta(v_1,\ldots,v_n)$$

Again, the coefficients in the last expression are possible values of ζ. Thus, α^* is equationally characterized. □

Theorem 4.6 (Completeness)**.** The axiom system F_1 is complete.

Proof. Let $\alpha = \beta$ be a valid equation. By Lemma 4.5, α and β are equationally characterized by $\alpha_1,\ldots,\alpha_n$ and $\beta_1,\ldots,\beta_m$. Using Lemma 4.4, we obtain the equation

$$\vdash_{F_1} (\alpha,\beta) = (\alpha_1,\beta_1) = \sum_{j=1}^{r} (\alpha_{1j},\beta_{1j})x_j + (\delta(\alpha),\delta(\alpha))$$

and define the set $M_1 \coloneqq \{(\alpha,\beta)\} \cup \{(\alpha_{1j},\beta_{1j}) \mid j \leq r\}$. We apply this equation on each pair in M_1, resulting in

$$\forall(\alpha',\beta') \in M_1\colon\ \vdash_{F_1} (\alpha',\beta') = \sum_{j=1}^{r} (\alpha'_{1j},\beta'_{1j})x_j + (\delta(\alpha'),\delta(\alpha')).$$

Next, we define $M_2 \coloneqq M_1 \cup \{(\alpha'_{1j}, \beta'_{1j}) \mid j \leq r\}$. This procedure can be reapplied until we have constructed a set that adds no additional pairs, which we call M'. Note that M' can have at most nm elements.

Finally, the equations

$$\forall(\alpha', \beta') \in M' \colon \vdash_{F_1} (\alpha', \beta') = \sum_{j=1}^{r} (\alpha'_j, \beta'_j) x_j + (\delta(\alpha'), \delta(\alpha'))$$

can be rewritten as

$$\forall(\alpha', \beta') \in M' \colon \vdash_{F_1} (\alpha', \beta') = \sum_{(\alpha'', \beta'') \in M'} (\alpha'', \beta'') \gamma(\alpha'') + (\delta(\alpha'), \delta(\alpha')),$$

where $\gamma(\alpha'')$ is either 0 or $x_{v_1} + \ldots + x_{v_h}$ for $h \geq 1$ and $1 \leq v_1 < \cdots < v_h \leq r$. Thus, $\gamma(\alpha'')$ never possesses empty word property. By Lemma 4.2, we infer

$$\forall(\alpha', \beta') \in M' \colon \vdash_{F_1} \alpha' = \beta'$$

and especially $\vdash_{F_1} \alpha = \beta$. □

4.2 Kleene Algebra

In this section we discuss Kleene algebras, whose properties were studied by Dexter Kozen in [5].

For regular expressions α and β, we recall the notation $\alpha \leq \beta$ as

$$\alpha \leq \beta \iff \alpha + \beta = \beta.$$

Moreover, if $\alpha \leq \beta$ is a valid equation, it is easy to see that $|\alpha| \subseteq |\beta|$.

Definition 4.7. A *Kleene algebra* is an algebraic structure $\mathcal{K} = (K, +, \cdot, ^*, 0, 1)$ that satisfies the following identities:

$$\alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma \tag{A+}$$
$$\alpha(\beta\gamma) = (\alpha\beta)\gamma \tag{A$\cdot$}$$
$$\alpha + \beta = \beta + \alpha \tag{C+}$$
$$\alpha(\beta + \gamma) = \alpha\beta + \alpha\gamma \tag{DL}$$
$$(\alpha + \beta)\gamma = \alpha\gamma + \beta\gamma \tag{DR}$$

$$\alpha + \alpha = \alpha \tag{I}$$
$$\alpha + 0 = \alpha \tag{U1}$$
$$\alpha \cdot 1 = 1 \cdot \alpha = \alpha \tag{U2}$$
$$\alpha \cdot 0 = 0 \cdot \alpha = 0 \tag{U3}$$

$$1 + \alpha\alpha^* \leq \alpha^* \tag{12}$$
$$1 + \alpha^*\alpha \leq \alpha^* \tag{13}$$

In addition to the aforementioned axioms, the Kleene algebra satisfies the following transformations:

$$\frac{\alpha\beta \leq \beta}{\alpha^*\beta \leq \beta} \tag{K$_\text{l}$}$$

$$\frac{\beta\alpha \leq \beta}{\beta\alpha^* \leq \beta} \tag{K$_\text{r}$}$$

Theorem 4.8. The theory of Kleene algebras is sound.

Proof. The axioms (A+) to (U3) are among the identities proven in chapter 2. Equations (12) and (13) can be proven by combining the valid identities $1+\alpha\alpha^* = \alpha^*$, $1 + \alpha^*\alpha = \alpha^*$ and $\alpha + \alpha = \alpha$. In order to verify K$_\text{l}$, note that if $\alpha\beta \leq \beta$ is valid, then $|\alpha\beta| \subseteq |\beta|$. Through induction, we can prove $|\alpha^*\beta| \subseteq |\beta|$ and thus K$_\text{l}$ and, by symmetry, K$_\text{r}$ are valid. □

Unlike the system F_1, Kleene algebras are not limited to regular expressions. One important type of Kleene algebras, and one which we need in order to prove the completeness of Kleene algebras, is defined over matrices.

Theorem 4.9. Let $\mathcal{K}$ be a Kleene algebra and Q be a finite set of indices. Furthermore, let Z_Q be the zero matrix and I_Q be the identity matrix in $\mathcal{K}^{Q\times Q}$. Then the structure

$$(\mathcal{K}^{Q\times Q}, +, \cdot, ^*, Z_Q, I_Q)$$

is a Kleene algebra.

Proof. See [5]. □

Definition 4.10. A *finite automaton* over $\mathcal{K}$ with states Q is a triple

$$\mathcal{U} \coloneqq (a, U, b),$$

where $a, b \in \{0,1\}^Q$ and $U \in \mathcal{K}^{Q\times Q}$. The vector a determines the *start states*, i.e. $q \in Q$ is a start state if and only if $a_q = 1$. Likewise, b defines the *final states*. Moreover, the $Q \times Q$ matrix U is called the *transition matrix*. The *language accepted by* $\mathcal{U}$ is defined as

$$a^\top U^* b.$$

Definition 4.11. Let $\mathcal{U} \coloneqq (a, U, b)$ be a finite automaton over a Kleene algebra $\mathcal{K}$. Then $\mathcal{U}$ is *simple* if there exist matrices $J, U_i \in \{0,1\}^{Q\times Q}$ for all $i \in A$ such that

$$U = J + \sum_{i\in A} i \cdot U_i.$$

If $J = 0$, then $\mathcal{U}$ is called *ε-free*. Additionally, $\mathcal{U}$ is *deterministic* if it is simple, ε-free and the vector a as well as each row in A_i have exactly one 1.

Lemma 4.12. Let X be a regular expression over the alphabet A. Then there exists a simple automaton $\mathcal{U} \coloneqq (a, U, b)$ such that

$$X = a^\top U^* b$$

Proof. Firstly, each one-symbol expression $x \in A \cup \{0,1\}$ can be expressed as the automaton

$$(\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 & x \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}),$$

since

$$\begin{bmatrix} 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & x \\ 0 & 0 \end{bmatrix}^* \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & x \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = x$$

Let (a, U, b) and (c, V, d) be simple automata such that

$$\alpha = a^\top U^* b$$
$$\beta = c^\top V^* d.$$

The expression $\alpha + \beta$ is accepted by the automaton

$$\left(\left[\begin{array}{c} a \\ \hline c \end{array}\right], \left[\begin{array}{c|c} U & 0 \\ \hline 0 & V \end{array}\right], \left[\begin{array}{c} b \\ \hline d \end{array}\right] \right),$$

as

$$\begin{aligned} \left[\begin{array}{c|c} a^\top & c^\top \end{array}\right] \cdot \left[\begin{array}{c|c} U & 0 \\ \hline 0 & V \end{array}\right]^* \cdot \left[\begin{array}{c} b \\ \hline d \end{array}\right] &= \left[\begin{array}{c|c} a & c \end{array}\right] \cdot \left[\begin{array}{c|c} U^* & 0 \\ \hline 0 & V^* \end{array}\right] \cdot \left[\begin{array}{c} b \\ \hline d \end{array}\right] \\ &= a^\top U^* b + c^\top V^* d \\ &= \alpha + \beta. \end{aligned}$$

Furthermore, $\alpha\beta$ is accepted by

$$\left(\left[\begin{array}{c} a \\ \hline 0 \end{array}\right], \left[\begin{array}{c|c} U & bc^\top \\ \hline 0 & V \end{array}\right], \left[\begin{array}{c} 0 \\ \hline d \end{array}\right] \right)$$

since

$$\begin{aligned} \left[\begin{array}{c|c} a^\top & 0 \end{array}\right] \cdot \left[\begin{array}{c|c} U & bc^\top \\ \hline 0 & V \end{array}\right]^* \cdot \left[\begin{array}{c} 0 \\ \hline d \end{array}\right] &= \left[\begin{array}{c|c} a^\top & 0 \end{array}\right] \cdot \left[\begin{array}{c|c} U^* & U^* bc^\top V^* \\ \hline 0 & V^* \end{array}\right] \cdot \left[\begin{array}{c} 0 \\ \hline d \end{array}\right] \\ &= a^\top U^* bc^\top V^* d \\ &= \alpha\beta. \end{aligned}$$

In order to construct an automaton for α^*, note that the following equations can be derived from the Kleene axioms, as proven in [5]:

$$\alpha^* = 1 + \alpha\alpha^* \tag{14}$$
$$(\alpha\beta)^*\alpha = \alpha(\beta\alpha)^* \tag{15}$$
$$(\alpha + \beta)^* = \alpha^*(\beta\alpha^*)^* \tag{S$_l$}$$

The automaton

$$(a, U + ba^\top, b)$$

accepts $\alpha\alpha^*$, since

$$\begin{aligned} a^\top (U + ba^\top)^* b &=^{(S_l)} a^\top U^* (ba^\top U^*)^* b \\ &=^{(15)} a^\top U^* b (a^\top U^* b)^* \\ &= \alpha\alpha^*. \end{aligned}$$

Finally, we can use the constructions for $\alpha + \beta$ and 1 to define an automaton for $1 + \alpha\alpha^* = \alpha^*$. □

Lemma 4.13. Let $\mathcal{U} \coloneqq (a, U, b)$ be a simple automaton. Then there exists a deterministic automaton $\mathcal{V} \coloneqq (c, V, d)$ such that

$$a^\top U^* b = c^\top V^* d.$$

Proof. First, we transform $\mathcal{U}$ into an ε-free automaton that accepts the same language. According to Definition 4.11, the matrix U of a simple automaton can be written as $J + U'$, where $U' = \sum_{i \in A} i \cdot U_i$. Then

$$a^\top U^* b = a^\top (J + U')^* b =^{(\mathrm{S}_1)} a^\top J^* (U' J^*)^* b,$$

which allows us to define an ε-free variant $\mathcal{U}'$ of $\mathcal{U}$ as

$$\mathcal{U}' \coloneqq \left((a^\top J^*)^\top, U' J^*, b \right).$$

For the remaining proof, assume that $\mathcal{U}$ is ε-free.

Let Q be the set of states used in $\mathcal{U}$. Using the notation $\mathcal{P}(Q)$ for the power set of Q, we define the canonical vector e_s for $s \in P(Q)$, i.e. $e_s \in \{0,1\}^{\mathcal{P}(Q)}$ such that e_s has the term 1 in position s and 0 otherwise. Define the matrix $\Phi \in \{0,1\}^{\mathcal{P}(Q) \times Q}$ such that

$$e_s^\top \Phi = s^\top$$

for each $s \in \mathcal{P}(Q)$. Furthermore, for each $i \in A$ let $V_i \in \{0,1\}^{\mathcal{P}(Q) \times \mathcal{P}(Q)}$ such that

$$e_s^\top V_i = e_{\left(s^\top U_i\right)}$$

for each $s \in \mathcal{P}(Q)$. Then

$$\begin{aligned} c &= e_a \\ V &= \sum_{i \in A} a \cdot V_i \\ d &= \Phi b \end{aligned}$$

The matrix Φ satisfies the equation

$$\Phi U = V \Phi,$$

as

$$\begin{aligned} e_s^\top \Phi U &= s^\top U \\ &= \sum_{i \in A} i \cdot s^\top U_i \\ &= \sum_{i \in A} i \cdot e_{s^\top U_i} \Phi \\ &= \sum_{i \in A} i \cdot e_s^\top V_i \Phi \\ &= e_s^\top V \Phi. \end{aligned}$$

It can be shown that Kleene algebras satisfy the transformation

$$\frac{x\alpha = \beta x}{x\alpha^* = \beta^* x}$$

and that this relation can be extended to matrices [5]. Thus, we conclude

$$\Phi U^* = V^* \Phi$$

and consequently

$$c^\top V^* d = e_a^\top V^* \Phi b = e_a^\top \Phi U^* b = a^\top U^* b.$$

□

Definition 4.14. Let $\mathcal{U} \coloneqq (a, U, b)$ be a deterministic automaton with states Q. For $q \in Q$, we define $e_q \in \{0,1\}^Q$ as the canonical vector of q, i.e. the vector with 1 in position q and 0 otherwise. A state $q \in Q$ is *reachable* if and only if

$$a^\top U^* e_q \neq 0.$$

Furthermore, assuming that each state in Q is reachable, we define the relation $\sim$ over Q such that $p \sim q$ implies the conditions

$$\begin{aligned} \forall i \in A\colon\ \delta(p,i) &= \delta(q,i) \\ e_p^\top b &= e_q^\top b. \end{aligned}$$

We also define

$$\begin{aligned} [p] &\coloneqq \{q \in Q \mid q = p\} \\ Q/_\sim &\coloneqq \{[p] \mid p \in Q\}. \end{aligned}$$

Lemma 4.15. Let $\mathcal{U} \coloneqq (a, U, b)$ be a deterministic automaton with states Q. Then there exists a deterministic automaton $\mathcal{V} \coloneqq (c, V, d)$ with states $Q/_\sim$ such that every state is reachable and

$$a^\top U^* b = c^\top V^* d.$$

We call $\mathcal{V}$ a *minimal deterministic automaton*.

Proof. First, we want to remove all unreachable states from $\mathcal{U}$. The matrix U can be written as

$$U = \sum_{i \in A} i \cdot U_i.$$

Furthermore, for each $q \in Q$ and $i \in A$, $\delta(q,i)$ denotes a state that satisfies

$$e_p^\top U_i = e_{\delta(q,i)}^\top.$$

The state $\delta(q,i)$ exists and is unique because $\mathcal{U}$ is deterministic. Define $R \subseteq Q$ as the set of all reachable states and $S = Q \setminus R$. Using this partition, we split the elements in (a, U, b) into subvectors and submatrices:

$$a = \left[\begin{array}{c} a_R \\ \hline a_S \end{array}\right], U = \left[\begin{array}{c|c} U_{RR} & U_{RS} \\ \hline U_{SR} & U_{SS} \end{array}\right], b = \left[\begin{array}{c} b_R \\ \hline b_S \end{array}\right]$$

The submatrix U_{RS} is 0, as unreachable states cannot be reached from states in R. Likewise, $a_S = 0$. Hence,

$$\begin{aligned} a^\top U^* b &= \left[\begin{array}{c|c} a_R^\top & 0 \end{array}\right] \cdot \left[\begin{array}{c|c} U_{RR} & 0 \\ \hline U_{SR} & U_{SS} \end{array}\right]^* \cdot \left[\begin{array}{c} b_R \\ \hline b_S \end{array}\right] \\ &= \left[\begin{array}{c|c} a_R^\top & 0 \end{array}\right] \cdot \left[\begin{array}{c|c} U_{RR}^* & 0 \\ \hline U_{SS}^* U_{SR} U_{RR}^* & U_{SS}^* \end{array}\right] \cdot \left[\begin{array}{c} b_R \\ \hline b_S \end{array}\right] \\ &= a_R^\top U_{RR}^* b_R. \end{aligned}$$

Assume that each state in Q is reachable. We define the relation $\sim$ as in Definition 4.14 and $e_{[p]}$ as the canonical vector for $[p] \in Q/\!\sim$. Let $\Psi \in \{0,1\}^{Q \times Q/\sim}$ such that

$$e_p^\top \Psi = e_{[p]}^\top$$

for each $p \in Q$. Furthermore, for each $i \in A$, define $V_i \in \{0,1\}^{Q/\sim \times Q/\sim}$ and $d \in \{0,1\}^{Q/\sim}$ such that

$$\begin{aligned} e_{[p]}^\top V_i &= e_{[\delta(p,i)]}^\top \\ e_{[p]}^\top d &= e_p^\top b, \end{aligned}$$

which implies

$$\Psi d = b.$$

Lastly, let

$$\begin{aligned} c^\top &= a^\top \Psi \\ V &= \sum_{i \in A} i \cdot V_i. \end{aligned}$$

Similarly to the matrix Φ in Lemma 4.13, it can be shown that Ψ satisfies the equation

$$U\Psi = \Psi V$$

and subsequently

$$U^*\Psi = \Psi V^*.$$

Therefore,

$$c^\top V^* d = a^\top \Psi V^* d = a^\top U^* \Psi d = a^\top U^* b.$$

□

Theorem 4.16 (Completeness)**.** Let $\alpha = \beta$ be a valid equation of regular expressions. Then $\alpha = \beta$ is a theorem of the theory of Kleene algebras.

Proof. By Lemma 4.12, 4.13 and 4.15, there exist minimal deterministic automata $\mathcal{U} \coloneqq (a, U, b)$ and $\mathcal{V} \coloneqq (c, V, d)$ such that

$$\begin{aligned} \alpha &= a^\top U^* b \\ \beta &= c^\top V^* d. \end{aligned}$$

Since minimal automata that accept the same language are isomorphic, there exists a permutation matrix P such that

$$\begin{aligned} U &= P^\top V P \\ a &= P^\top c \\ b &= P^\top d. \end{aligned}$$

Hence,

$$\begin{aligned} \alpha &= a^\top U^* b \\ &= (P^\top c)^\top (P^\top V P)^* (P^\top d) \\ &= c^\top P (P^\top V P)^* P^\top d \\ &= c^\top P P^\top V^* P P^\top d \\ &= c^\top V^* d \\ &= \beta. \end{aligned}$$

□

5 Proof Complexity

In the previous chapters, we construct axiom systems that generate proofs for equations of regular expressions. However, the problem of proving that an equation is valid can also be examined from a complexity perspective. An important idea in this regard is Cook's program, which links the equivalence of NP and coNP to polynomially bounded propositional proof systems [2]. We will define a similar statement that is based on regular expressions instead of propositional logic. Furthermore, we will show a link between regular expressions and the equivalence of NP and PSPACE. In this chapter, we follow the proofs given by Simon Beier and Markus Holzer in [1] as well as the proof by Dexter Kozen in [4].

We define the problem EQUIV as

$$\text{EQUIV} \coloneqq \{(X, Y) \mid X \text{ and } Y \text{ are regular expressions such that } |X| = |Y|\}.$$

Definition 5.1. The class NP consists of all languages that can be decided by a nondeterministic Turing machine in polynomial time. Furthermore, PSPACE is the set of all problems that can be decided by a deterministic Turing machine on a tape of polynomial length. By Savitch's Theorem [9], the definition of PSPACE can be extended to nondeterministic Turing machines as well. Additionally, the class coNP consists of the complements of all elements in NP.

Definition 5.2. Let A be an alphabet and $L \subseteq A^*$. A *proof system* for L is a function $f : A^* \to L$ that is surjective and computable in deterministic polynomial time. For $x, y \in L$, we call x a *proof* of y if $y = f(x)$. Furthermore, f is *polynomially bounded* if there exists a polynomial function p such that each y has a proof x that satisfies $|x| \leq p(|y|)$, where $|.|$ denotes the length of a given word.

Lemma 5.3. Let $L \subseteq A^*$ and $L \neq \emptyset$. Then $L \in$ NP if and only if L has a polynomially bounded proof system.

Proof. If $L \in$ NP, there exists a nondeterministic Turing machine $\mathcal{U}$ that accepts L in polynomial time. Hence, if x encodes a computation that accepts y we can simply define $f(x) = y$. Otherwise, we set $f(x) = y_0$ for some $y_0 \in L$.

Conversely, if L has a polynomially bounded proof system, then y has a polynomially bounded proof x. Thus, a nondeterministic algorithm can compute x in polynomial time by simply guessing. □

Theorem 5.4. EQUIV is PSPACE-complete.

Proof. First, we show that EQUIV is in PSPACE. Let $X = Y$ be a valid equation of regular expressions. As shown in Lemma 4.12, we can construct nondeterministic finite automata $\mathcal{U}$ and $\mathcal{V}$ that accept the sets $|X|$ and $|Y|$ respectively. We construct a nondeterministic Turing machine to check whether the sets accepted by $\mathcal{U}$ and $\mathcal{V}$ are identical. To do this, the algorithm rejects if it finds a word w that is in $|X|$ but not in $|Y|$ or vice versa. It follows from Savitch's Theorem [9] that the classes PSPACE and NPSPACE are identical, which allows the algorithm to choose a word w nondeterministically. The procedure starts by marking each start state of $\mathcal{U}$ and $\mathcal{V}$. It then iterates through the word w symbol by symbol and moves the marks to all

states that are currently accessible by the string read so far. The algorithm rejects if there is a marked accept state on one automaton but no marked accept state on the other. This whole process can be achieved within polynomial space, so EQUIV is in PSPACE.

In order to show that EQUIV is PSPACE-hard, we take a fixed arbitrary problem in PSPACE. There exists a one-tape polynomial-space-bound deterministic Turing machine M over the alphabet $A \coloneqq \{0, 1, \textvisiblespace, \triangleright\}$ that decides this problem. Furthermore, let Q be the finite set of states used in M. For an input string x, let $n \coloneqq |x|$ and $c \geq 1$ be a constant such that n^c is the space bound of M. We are interested in the computation history of M, which we encode as a string containing a list of configurations α_i of length n^c:

$$\alpha_0 \alpha_1 \cdots \alpha_m$$

Each configuration $\alpha_i \coloneqq (q, u, v)$ is encoded as the string $qu\square v$, where $\square$ is an additional symbol used to mark the position of the cursor. We construct a regular expression Y that contains all strings which are *not* valid computation histories for an input x. Let $\Delta \coloneqq A \cup Q \cup \{\square\}$, which is the set of all symbols that can appear in a computation history. If M accepts x, then there exists a valid computation and $|Y| \neq \Delta^*$. Otherwise, all computations are invalid and Y describes the whole set Δ^*. Thus, the construction of Y proves that EQUIV is PSPACE-hard.

A computation history is invalid if it satisfies at least one of the following properties:

1. There exists $i \leq m$ such that $|\alpha_i| \neq n^c$ or the history contains no states at all.
2. A configuration contains more than one cursor or no cursor at all.
3. α_0 is not the start configuration.
4. α_m is not an accept configuration.
5. There exists $i \leq (m-1)$ such that the transition from α_i to $\alpha_{(i+1)}$ is not valid.

The regular expression Y can be defined as $Y \coloneqq Y_1 + Y_2 + Y_3 + Y_4 + Y_5$, where each Y_i encodes one of the properties above that invalidate a given computation history. Each term α_i starts with a symbol in Q, so Y_1 just has to check for occurrences of $q \in Q$ followed by more or less than n^c non-state symbols. Y_2 works in a similar manner. As there are only finitely many possible configurations of length n^c, the expressions Y_3 and Y_4 check whether α_0 or α_m consist of a string that does not satisfy the respective condition.

In order to encode the fifth rule, recall that the transition function δ of M can be defined as

$$\delta : Q \times A \to Q \times A \times \{\leftarrow, \rightarrow, -\}.$$

Thus, we can divide Y_5 into regular expressions Z_{qa} for each state $q \in Q$ and symbol $a \in A$. Each term Z_{qa} is satisfied if a configuration in the state q and a symbol a next to a cursor is followed by an incorrect configuration. The term Y_5 is then defined as the sum of all Z_{qa} for each $q \in Q$ and $a \in A$. □

Theorem 5.5. NP = PSPACE if and only if EQUIV admits a polynomially bounded proof system.

Proof. If NP = PSPACE, then EQUIV $\in$ NP. By Lemma 5.3, EQUIV must have a polynomially bounded proof system. Conversely, EQUIV having a polynomially bounded proof system implies EQUIV $\in$ NP. Since EQUIV is PSPACE-complete, we conclude that PSPACE $\subseteq$ NP and thus NP = PSPACE. □

The set $\text{EQUIV}_{\text{fin}}$ is a modification of EQUIV that contains pairs of regular expressions that only use unions and concatenations. Regular expressions in $\text{EQUIV}_{\text{fin}}$ always describe finite languages, which is why we use the subscript "fin" as in finite.

Theorem 5.6. NP = coNP if and only if $\text{EQUIV}_{\text{fin}}$ admits a polynomially bounded proof system.

Proof. Observe that $\text{EQUIV}_{\text{fin}}$ is in coNP. In order to prove that an equation is not in $\text{EQUIV}_{\text{fin}}$, we can verify that a given assignment refutes the equality in deterministic polynomial time. If NP = coNP, then $\text{EQUIV}_{\text{fin}} \in$ NP. By Lemma 5.3, $\text{EQUIV}_{\text{fin}}$ thus has a polynomially bounded proof system.

Conversely, $\text{EQUIV}_{\text{fin}}$ having a polynomially bounded proof system implies $\text{EQUIV}_{\text{fin}} \in$ NP. It can be shown that $\text{EQUIV}_{\text{fin}}$ is coNP-complete [3], which means that each problem in coNP can be reduced to $\text{EQUIV}_{\text{fin}}$. Thus, we infer coNP $\subseteq$ NP.

Furthermore, the complementary set $\text{EQUIV}^c_{\text{fin}}$, i.e. the set of all equations of finitary regular expressions that are not valid, is NP-complete. As $\text{EQUIV}_{\text{fin}} \in$ NP, it follows that $\text{EQUIV}^c_{\text{fin}} \in$ coNP. Thus, each problem in NP can be reduced to $\text{EQUIV}^c_{\text{fin}}$, which implies NP $\subseteq$ coNP. □

References

[1] Simon Beier and Markus Holzer. On regular expression proof complexity of Salomaa's axiom system F_1. In Stefan Kiefer, Jan Kretínský, and Antonín Kucera, editors, *Taming the Infinities of Concurrency - Essays Dedicated to Javier Esparza on the Occasion of His 60th Birthday*, volume 14660 of *Lecture Notes in Computer Science*, pages 72–100. Springer, 2024.

[2] Samuel R. Buss. Towards np-p via proof complexity and search. *Annals of Pure and Applied Logic*, 163(7):906–917, 2012.

[3] Harry B. Hunt III, Daniel J. Rosenkrantz, and Thomas G. Szymanski. On the equivalence, containment, and covering problems for the regular and context-free languages. *J. Comput. Syst. Sci.*, 12(2):222–268, 1976.

[4] Dexter Kozen. Introduction to Kleene algebra. Lecture notes, Spring term 2004, Cornell University, available at www.cs.cornell.edu/courses/cs786/2004sp/.

[5] Dexter Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. In *Proceedings of the Sixth Annual Symposium on Logic in Computer Science (LICS '91), Amsterdam, The Netherlands, July 15-18, 1991*, pages 214–225. IEEE Computer Society, 1991.

[6] Jacques Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009.

[7] Arto Salomaa. Two complete axiom systems for the algebra of regular events. *J. ACM*, 13(1):158–169, 1966.

[8] Arto Salomaa. *Theory of Automata*. Pergamon Press, 1969.

[9] Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. Syst. Sci.*, 4(2):177–192, 1970.