Learning Language Equations and Regular Languages using Alternating Finite Automata[*]

Aziz Fellah and Ajay Bandi
School of Computer Science and Information Systems
Northwest Missouri State University
Maryville, MO 64468
{afellah,ajay}@nwmissouri.edu

Abstract

Learning regular languages using several classes of finite state machines and different learning framework has been one of the objective in the theory of computation and computability courses. Alternating finite automata (AFA) are an appealing abstraction and a key ingredient in modeling many software systems and parallel computations. Despite the fact that AFA accept regular languages, they have several interesting properties due to their transition structures such as their mode of accepting is quite different from that of deterministic finite automata (DFA) and nondeterministic finite automata (NFA); and importantly, AFA are double-exponentially more succinct than DFA. This paper presents a new paradigm to learning regular languages and solving language equations using AFA. Such models can be described naturally as a set of equations that parallels the solutions of algebraic equations. Moreover, the solution of such systems of equations is the class of regular languages.

1 Introduction

Despite the rapid pace of the technology which has significantly altered many aspects of the CS programs, the theory of computation and its cluster of related disciplines continue to play a foundational role in the field of computer

[*]Copyright ©2019 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

science. Most undergraduate CS programs offer a course in theory of computation, alongside with formal languages, automata, and computability. Such a course expose students to different types of abstract models and the foundations of computation. Regular languages is an important class of formal languages in the Chomsky's hierarchy and are the first concepts to be taught in the theory of computation and computability courses. A regular language can be expressed symbolically with a regular expression, a regular grammar, and conventional models, namely deterministic finite automata (DFA) and nondeterministic finite automata (NFA). A vast amount of literature is devoted to learning formal and regular languages [10] which are used in parsing and designing programming languages. The concept of alternation [5, 9, 7] extends the notion of nondeterminism by including the universal quantifiers in addition to the existential quantifiers. A formalization of this concept yields the definition of alternating finite automata (AFA) which form a powerful abstraction layer beyond nondeterminism. AFA, a generalization of NFA, become an influential model that has received significant interests inside and outside the classroom, from instructors and researchers [11, 6, 9, 7, 8, 3, 1]. Similarly to DFA and NFA, AFA are equally expressive as the class of regular languages. Aside from the applications in modeling many systems and phenomena, AFA models capture different level of abstractions that are used in software system designs, model checking, and various areas of computer science [11, 6, 8]. While all these formalisms, DFA, NFA and AFA, are equivalent in expressive power terms of language recognition, they differ in other terms. Importantly, NFA are exponentially more succinct than DFA but AFA are double-exponentially more succinct than DFA. Language equations are equations over language operations where both the constants and variables are languages. Usually, they are formalized through various classes of finite state machines such as DFA and NFA using two operations, union and concatenation. The relevant properties of language equations, such that existence and uniqueness of their solutions have been established in the literature [7, 4].

An *alphabet* is a finite, nonempty set. The elements of an alphabet are called *symbols* or *letters*. A *word* (string) over an alphabet Σ is a finite sequence consisting of zero or more letters of Σ. The word consisting of zero letters is called the *empty word*, denoted by ϵ. By definition, $|\epsilon| = 0$. The set of all words (respectively all nonempty words) over an alphabet Σ is denoted by Σ^* (Σ^+). Given a language L over an alphabet Σ, the Kleene closure (Kleene star "*") of L is the set $L^* = \bigcup_{i=0}^{\infty} L^i$ and the positive Kleene closure (Kleene plus "+") of L is $L^+ = \bigcup_{i=1}^{\infty} L^i$. The language $\overline{L} = \Sigma^* \setminus L$ is the complement of L. The *concatenation* of two words w_1 and w_2 is the word consisting of the symbols of w_1 followed by the symbols of w_2, denoted $w_1 \cdot w_2$. The length of a word w, denoted by $|w|$, is the number of letters in w. We denote by the symbol $\mathcal{B}$ the Boolean semiring, $\mathcal{B} = \{0, 1\}$. Let Q be a set. Then $\mathcal{B}^Q$ is the set of all mappings of Q into $\mathcal{B}$. Note that $u \in \mathcal{B}^Q$ can also be considered as a Q-vector over $\mathcal{B}$.

The remaining of the paper is organized as follows. In section 2, we describe alternating finite automata and state some definitions and results that can be used in subsequent sections. Section 3 introduces systems of equations to represent AFA. These equations, and not the transition diagrams, are in essence the underlying algebraic framework that makes AFA appealing from the point of view of representation and flexibility. In section 4, we exhibit constructions for the Boolean operations on AFA. Section 5, shows such systems of equations parallel the solution of algebraic equations. In particular, the solution of such systems of equations is the class of regular languages. We conclude the paper with some potential discussions in section 6. This paper is based on our original work on AFA but summarized in a compact form given the limitation on the conference page number.

2 Alternating Finite Automata

NFA are a generalization of DFA by allowing a state to have multiple outgoing transitions labeled with the same symbol or a ϵ-transition. A string is accepted by an NFA if there exists some path that leads to an accepting state. In a nondeterministic computation all configurations are *existential* in the sense that there exists at least one successful path that leads to acceptance. An alternating finite automaton (AFA) may have also *universal* configurations from which the computation branches into a number of parallel computations that must all lead to acceptance. We represent existential and universal choices by a Boolean formula. Formally, let Q be a set, we use $\mathcal{B}^Q$ to be the set of all Boolean formulas over Q. That is, $\mathcal{B}^Q$ is built from the elements $q \in Q$, 1, and 0 using the binary operations *and* ($\vee$), *or* ($\wedge$), and *not* ($^-$). If in a given state q the AFA reads an input symbol a, it will activate all states of the AFA to work on the remaining part of the input in parallel. Once the states have completed their tasks, q will evaluate their results using a Boolean function and pass on the resulting value to the state by which it was activated. A word w is accepted if the starting state computes the value of 1. Otherwise, it is rejected. We now formalize this idea.

Definition 1 *An alternating finite automaton* (AFA) *is a quintuple* $A = (Q, \Sigma, s, F, g)$ *where* (a) Q *is a finite set, the set of states;* (b) Σ *is an alphabet, the input alphabet;* (c) $s \in Q$ *is the starting state;* (d) $F \subseteq Q$ *is the set of final states;* (e) g *is a mapping of* Q *into the set of all mappings of* $\Sigma \times \mathcal{B}^Q$ *into* $\mathcal{B}$.

We turn to defining the sequential behavior of an AFA. For $q \in Q$ and $a \in \Sigma$, let $g_q(a)$ be the Boolean function defined as:

$$g_q(a, u) : \Sigma \times \mathcal{B}^Q \to \mathcal{B}$$

where $a \in \Sigma$ and $u \in \mathcal{B}^Q$. Also, for $a \in \Sigma$, $q \in Q$, and $u \in \mathcal{B}^Q$, $g_q(a, u) = g_q(a)(u)$, is equals to either 0 or 1. Now define $f \in \mathcal{B}^Q$ by the condition

$$f_q = 1 \iff q \in F.$$

f is called the *characteristic vector* of F. We extend g to a mapping of Q into the set of all mappings of $\Sigma^* \times \mathcal{B}^Q$ into $\mathcal{B}$ as follows:

$$g_q(w, u) = \begin{cases} u_q & \text{if } w = \epsilon, \\ g_q(a, g(v, u)) & \text{if } w = av \text{ with } a \in \Sigma \text{ and } v \in \Sigma^* \end{cases}$$

where $w \in \Sigma^*$ and $u \in \mathcal{B}^Q$.

Definition 2 *Let $A = (Q, \Sigma, s, F, g)$ be an AFA. A word $w \in \Sigma^*$ is accepted by A if and only if $g_s(w, f) = 1$. The language accepted by A is the set $L(A) = \{w \mid w \in \Sigma^* \ \wedge \ g_s(w, f) = 1\}$.*

Example 1 Consider the following AFA $A = (Q, \Sigma, s, F, g)$ where $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{a, b\}$, $s = \{q_0\}$, $F = q_2$, and g is given by the following table. The example shows a run of the AFA on the input *bab*.

State	a	b
q_0	$q_0 \wedge q_1$	$q_1 \vee \overline{q_2}$
q_1	q_0	$\overline{q_0} \wedge q_2$
q_2	$q_0 \vee \overline{q_1}$	1

In the same example of AFA, we have drawn the existential states as $\vee$ and the universal states as $\wedge$. The AFA can have multiple runs on a given input where both of these choices coexist. Notice that the run branches in parallel to two states, $\overline{q}_0$ and q_2 on the second input symbol b from q_1.

Example 2 Let $w = bab$ be a string. We will check whether w is accepted by the NFA A.

$$\begin{aligned}
& g_{q_0}(bab, f) \\
= \ & g_{q_1}(ab, f) \vee \overline{g_{q_2}(ab, f)} \\
= \ & g_{q_0}(b, f) \vee \overline{(g_{q_0}(b, f) \vee \overline{g_{q_1}(b, f)})} \\
= \ & (g_{q_1}(\epsilon, f) \vee \overline{g_{q_2}(\epsilon, f)}) \vee \overline{((g_{q_1}(\epsilon, f) \vee \overline{g_{q_2}(\epsilon, f)}) \vee (\overline{\overline{\overline{g_{q_0}(\epsilon, f)} \wedge g_{q_2}(\epsilon, f)}}))} \\
= \ & (0 \vee \overline{1}) \vee \overline{((0 \vee \overline{1}) \vee \overline{(\overline{0} \wedge 1)})} = (0 \vee 0) \vee \overline{((0 \vee 0) \vee \overline{(1 \wedge 1)})} \\
= \ & (0) \vee \overline{((0) \vee \overline{(1)})} = 0 \vee \overline{(0 \vee 0)} = 0 \vee \overline{(0)} = 0 \vee 1 \\
= \ & 1
\end{aligned}$$

Therefore the string *bab* is accepted.

Proposition 1 [9, 4] *AFA are equivalent in term of language recognition power to NFA and DFA. That is, they can only accept regular languages.*

Proposition 2 *AFA are double-exponentially more succinct that DFA.*

Let $A = (Q, \Sigma, S, F, g)$ be an AFA to simulate an equivalent DFA $A' = (Q', \Sigma, S', F', g')$, Q' needs $2^{2^{|Q|}}$ Boolean functions on Q. For instance, an NFA may be exponentially smaller than the minimal DFA, and an AFA may be doubly-exponentially smaller than the minimal DFA.

3 Representing AFA by Systems of Language Equations

Language equations are equations defined over languages where both the constants and variables are formal languages. It is well-known that regular languages can be described as the solutions of systems of one-sided linear equations in an appropriate semiring [7, 4, 2]. We show that AFA can be readily represented by systems of equations. However, the systems of equations to be considered involve Boolean expressions over a finite set X of variables and the symbols of an alphabet Σ. The main result of this section is that the solutions of such systems of equations are precisely the regular languages and that, indeed, there is a natural correspondence between AFA and such systems of equations. In the sequel we associate with each AFA a system of equations such that the languages accepted by the AFA with various start states constitute the unique fixpoint of the system of equations.

Let $A = (Q, \Sigma, s, F, g)$ be an AFA. For $q \in Q$, we use x_q to denote a boolean variable associated with the state q and $\overline{x}_q$ to denote its negation. Let $X_Q = \{x_q \mid q \in Q\}$. Then the following system $L(A)$ of equations can be used to describe A:

$$L(A) = \left\{ X_q = \sum_{a \in \Sigma} a \cdot g_q(a, X) + \epsilon(f_q) \right\}_{q \in Q} \tag{1}$$

$$\epsilon(f_q) = \begin{cases} \epsilon & \text{if } f_q = 1 \\ \emptyset & \text{otherwise} \end{cases}$$

In the system $L(A)$ of equations, the Boolean function $g_q(a, X)$ is considered as being given by a Boolean expression in $\mathcal{B}(X_Q)$. Any system of language equations of the above form has a unique solution for each X_Q, $q \in Q$. Furthermore, the solution for each X_Q is regular.

4 Boolean Operations on AFA

Of course, as every AFA language is regular, the class of AFA languages is closed under the Boolean operations. However, this is only a "surface" result. Rather than this type of existence result one would like to have concrete constructions for AFA.

Let $A = (Q, \Sigma, s, F, g)$ be an AFA. First we construct the complement of A

$$\overline{A} = (Q, \Sigma, s, F', g')$$

such that $L(\overline{A}) = \Sigma^* \setminus L(A)$. The set F' of final states is defined by the condition

$$q \in F' \iff \begin{cases} q \in F, & \text{if } q \neq s, \\ q \notin F, & \text{if } q = s, \end{cases}$$

for $q \in Q$. For $u \in \mathcal{B}^Q$, let u' be the mapping given by

$$u'_q = \begin{cases} u_q, & \text{if } q \neq s, \\ \overline{u}_q, & \text{if } q = s, \end{cases}$$

for $q \in Q$. The function g' is given by

$$g'_q(a, u) = \begin{cases} g_q(a, u'), & \text{if } q \neq s, \\ \overline{g_q(a, u)}, & \text{if } q = s, \end{cases}$$

where $q \in Q$, $a \in \Sigma$, and $u \in \mathcal{B}^Q$.

Theorem 1 $L(\overline{A}) = \Sigma^* \setminus L(A)$.

Proof: We prove that $g'(w, f') = g(w, f)'$ for all $w \in \Sigma^*$. This is obviously true for $w = \epsilon$. Now assume that the statement holds for $v \in \Sigma^*$ and consider $w = av$ with $a \in \Sigma$. For $q \in Q$, one has

$$g'_q(w, f') = g'_q(a, g(v, f')') = g'_q(a, g(v, f)') = g_q(w, f)'.$$

Thus, $g'_s(w, f') = \overline{g_s(w, f)}$, that is, $w \in L(\overline{A}) \iff w \notin L(A)$. $\square$

Example 3 Let the AFA $A = (Q, \Sigma, s, F, g)$, where $Q = \{1, 2, 3, 4\}$, $\Sigma = \{a, b\}$, $s = 1$, $F = \{1, 4\}$ and g is given as follows. A is represented by $L(A)$. We construct the complement of A, $\overline{A}$, that accepts $\overline{L(A)}$.

$$\begin{array}{llcl} L(A): & x_1 & = & a \cdot (\overline{x}_2 \vee \overline{x}_3) + b \cdot (\overline{x}_2 \vee \overline{x}_3 \vee \overline{x}_4) + \epsilon \\ & x_2 & = & a \cdot (x_2 \vee x_3 \wedge x_4) + b \cdot (x_2 \wedge x_3 \wedge x_4) \\ & x_3 & = & a \cdot (x_2 \wedge x_3 \vee x_3 \wedge \overline{x}_4 \vee \overline{x}_3 \wedge x_4) + b \cdot (x_2 \wedge x_3 \wedge x_4) \\ & x_4 & = & a \cdot (x_2 \wedge x_3 \vee x_2 \wedge \overline{x}_4 \vee x_3 \wedge \overline{x}_4) + b \cdot (x_2 \wedge x_3 \wedge x_4) + \epsilon \end{array}$$

$$\begin{array}{llcl} \overline{L(A)}: & x_1 & = & a\cdot(x_2\wedge x_3)+b\cdot(x_2\wedge x_3\wedge\overline{x}_4)\\ & x_2 & = & a\cdot(x_2\vee x_3\wedge\overline{x}_4)+b\cdot(x_2\wedge x_3\wedge\overline{x}_4)\\ & x_3 & = & a\cdot(x_2\wedge x_3\vee x_3\wedge x_4\vee\overline{x}_3\wedge\overline{x}_4)+b\cdot(x_2\wedge x_3\wedge\overline{x}_4)\\ & x_4 & = & a\cdot(x_2\wedge x_3\vee x_2\wedge x_4\vee x_3\wedge x_4)+b\cdot(x_2\wedge x_3\wedge\overline{x}_4)+\epsilon \end{array}$$

Our next construction is for the union of languages accepted by AFA. For $i=1,2$ let $A^{(i)}=(Q^{(i)},\Sigma,s^{(i)},F^{(i)},g^{(i)})$ be two AFA with disjoint state sets. We construct an AFA

$$A=A^{(1)}\vee A^{(2)}=(Q,\Sigma,s,F,g)$$

such that $L(A^{(1)}\vee A^{(2)})=L(A^{(1)})\cup L(A^{(2)})$. Let s be a new state symbol, $s\notin Q^{(1)}\cup Q^{(2)}$, let

$$Q=Q^{(1)}\cup Q^{(2)}\cup\{s\},$$

$$F=\begin{cases} F^{(1)}\cup F^{(2)}, & \text{if } s^{(1)}\notin F^{(1)} \text{ and } s^{(2)}\notin F^{(2)},\\ F^{(1)}\cup F^{(2)}\cup\{s\}, & \text{otherwise.}\end{cases}$$

The function g is given as follows

$$g_q(a,u)=\begin{cases} g_q^{(i)}(a,u\big|_{Q^{(i)}}), & \text{if } q\in Q^{(i)} \text{ with } i\in\{1,2\},\\ g^{(1)}_{s^{(1)}}(a,u\big|_{Q^{(1)}})\vee g^{(2)}_{s^{(2)}}(a,u\big|_{Q^{(2)}}), & \text{if } q=s.\end{cases}$$

where $q\in Q$, $a\in\Sigma$, and $u\in B^Q$.

Theorem 2 *$L(A^{(1)}\vee A^{(2)})=L(A^{(1)})\cup L(A^{(2)})$.*

Proof: By induction on $|w|$ one verifies that

$$g_q(w,f)=\begin{cases} g^{(1)}_{s^{(1)}}(w,f^{(1)})\vee g^{(2)}_{s^{(2)}}(w,f^{(2)}), & \text{if } q=s,\\ \\ g_q^{(i)}(w,f^{(i)}), & \text{if } q\in Q^{(i)} \text{ with } i\in\{1,2\},\end{cases}$$

for $q\in Q$ and $w\in\Sigma^*$. $\square$

The construction of an AFA

$$A=A^{(1)}\wedge A^{(2)}=(Q,\Sigma,s,F,g)$$

such that $L(A^{(1)}\wedge A^{(2)})=L(A^{(1)})\cap L(A^{(2)})$ is similar. With Q as above, one defines

$$F=\begin{cases} F^{(1)}\cup F^{(2)}, & \text{if } s^{(1)}\notin F^{(1)} \text{ or } s^{(2)}\notin F^{(2)},\\ F^{(1)}\cup F^{(2)}\cup\{s\}, & \text{otherwise.}\end{cases}$$

and

$$g_q(a,u) = \begin{cases} g_q^{(i)}(a, u\big|_{Q^{(i)}}), & \text{if } q \in Q^{(i)} \text{ with } i \in \{1,2\}, \\ \\ g^{(1)}_{s^{(1)}}(a, u\big|_{Q^{(1)}}) \wedge g^{(2)}_{s^{(2)}}(a, u\big|_{Q^{(2)}}), & \text{if } q = s. \end{cases}$$

where $q \in Q$, $a \in \Sigma$, and $u \in B^Q$. One then proves that $L(A)$ is indeed the intersection of the two languages.

Theorem 3 $L(A^{(1)} \wedge A^{(2)}) = L(A^{(1)}) \cap L(A^{(2)})$.

Example 4 Let the AFA $A^{(1)} = (Q^{(1)}, \Sigma, s^{(1)}, F^{(1)}, g^{(1)})$ where $Q^{(1)} = \{1,2,3,4\}$, $\Sigma = \{a,b\}$, $s^{(1)} = \{1\}$, $F^{(1)} = \{1,4\}$, and $g^{(1)}$ is given as follows.

$$\begin{array}{lrcl} L(A^{(1)}): & x_1 & = & a \cdot (\overline{x}_2 \vee \overline{x}_3) + b \cdot (\overline{x}_2 \vee \overline{x}_3 \vee \overline{x}_4) + \epsilon \\ & x_2 & = & a \cdot (x_2 \vee x_3 \wedge x_4) + b \cdot (x_2 \wedge x_3 \wedge x_4) \\ & x_3 & = & a \cdot (x_2 \wedge x_3 \vee x_3 \wedge \overline{x}_4 \vee \overline{x}_3 \wedge x_4) + b \cdot (x_2 \wedge x_3 \wedge x_4) \\ & x_4 & = & a \cdot (x_2 \wedge x_3 \vee x_2 \wedge \overline{x}_4 \vee x_3 \wedge \overline{x}_4) + b \cdot (x_2 \wedge x_3 \wedge x_4) + \epsilon \end{array}$$

Let the AFA $A^{(2)} = (Q^{(2)}, \Sigma, s^{(2)}, F^{(2)}, g^{(2)})$ where $Q^{(2)} = \{5,6,7\}$, $\Sigma = \{a,b\}$, $s^{(2)} = \{5\}$, $F^{(2)} = \{5,7\}$ and $g^{(2)}$ is given as follows.

$$\begin{array}{lrcl} L(A^{(2)}): & x_5 & = & a \cdot (x_5) + b \cdot (x_5 \wedge \overline{x}_6 \vee x_5 \wedge \overline{x}_7 \vee \overline{x}_5 \wedge x_6 \wedge x_7) + \epsilon \\ & x_6 & = & a \cdot (x_6) + b \cdot (x_6 \wedge \overline{x}_7 + \overline{x}_6 \wedge x_7) \\ & x_7 & = & a \cdot (x_7) + b \cdot (\overline{x}_7 \wedge x_5 \wedge x_6) + \epsilon \end{array}$$

$A^{(1)} \wedge A^{(2)} = (Q, \Sigma, s, F, g)$ where $Q = \{1,2,3,4,5,6,7\}$, $\Sigma = \{a,b\}$, $s = \{0\}$, $F = \{1,4,5,7\}$, $L(A^{(1)} \wedge A^{(2)}) = L(A^{(1)}) \cap L(A^{(2)})$, and g as follows.

$$\begin{array}{rcl} x_0 & = & a \cdot ((\overline{x}_2 \vee \overline{x}_3) \wedge x_5) + b \cdot ((\overline{x}_2 \vee \overline{x}_3 \vee \overline{x}_4) \wedge \\ & & (x_5 \wedge \overline{x}_6 \vee x_5 \wedge \overline{x}_7 \vee \overline{x}_5 \wedge x_6 \wedge x_7)) + \epsilon \\ x_1 & = & a \cdot (\overline{x}_2 \vee \overline{x}_3) + b \cdot (\overline{x}_2 \vee \overline{x}_3 \vee \overline{x}_4) + \epsilon \\ x_2 & = & a \cdot (x_2 \vee x_3 \wedge x_4) + b \cdot (x_2 \wedge x_3 \wedge x_4) \\ x_3 & = & a \cdot (x_2 \wedge x_3 \vee x_3 \wedge \overline{x}_4 \vee \overline{x}_3 \wedge x_4) + +b \cdot (x_2 \wedge x_3 \wedge x_4) \\ x_4 & = & a \cdot (x_2 \wedge x_3 \vee x_2 \wedge \overline{x}_4 \vee x_3 \wedge \overline{x}_4) + b \cdot (x_2 \wedge x_3 \wedge x_4) + \epsilon \\ x_5 & = & a \cdot (x_5) + b \cdot (x_5 \wedge \overline{x}_6 \vee x_5 \wedge \overline{x}_7 \vee \overline{x}_5 \wedge x_6 \wedge x_7) + \epsilon \\ x_6 & = & a \cdot (x_6) + b \cdot (x_6 \wedge \overline{x}_7 + \overline{x}_6 \wedge x_7) \\ x_7 & = & a \cdot (x_7) + b \cdot (\overline{x}_7 \wedge x_5 \wedge x_6) + \epsilon \end{array}$$

Other proofs are quite similar and omitted due to lack of space. A summary is as follows.

Proposition 3 *Language Equations are closed under concatenation, union, complement, intersection, Kleene star operations. That is,* $L(A^{(1)} \cdot A^{(2)}) = L(A^{(1)}) \cdot L(A^{(2)})$, $L(A^{(1)} \vee A^{(2)}) = L(A^{(1)}) \cup L(A^{(2)})$, $\overline{L(A)}$, $L(A^{(1)} \wedge A^{(2)}) = L(A^{(1)}) \cap L(A^{(2)})$, *and* $(L(A))^* = \bigcup_{i=0}^{\infty} (L(A))^i$.

5 Equations over Languages

Finite automata are represented as systems of language equations with two operations, union and concatenation. The relevant properties of these equations, such that existence and uniqueness of their solutions have been established in the literature. For instance, the equation $X = \alpha X + \beta$, where α and β are fixed languages, and it is well-known by Arden's Lemma that the equation has $\alpha^*\beta$ as solution and if the empty word $\epsilon \notin \alpha$, then it is the only solution.

Theorem 4 [7] *Let A be the equational representation of A. Let s be the starting state of A. Then the solution for X_s is exactly the language accepted by A. Furthermore, the system of equations of the form (1) has a unique solution for each $X_q \in Q$ and the solution for each X_q is regular.*

Example 5 Given the AFA $A = (Q, \Sigma, s, F, g)$, where $Q = \{1, 2, 3\}$, $\Sigma = \{a, b\}$, $s = 1$, $F = \{3\}$ and g is given by the following system of language equations. The regular language $L(A)$ generated by A is obtained by solving the following system of language equations. The subscript operators $*$ and $+$ indicate the Kleene star and Kleene plus operators, respectively.

$$\begin{aligned} x_1 &= a \cdot (x_1 \vee x_2) + b \cdot x_3 \\ x_2 &= a \cdot x_1 + b \cdot (x_1 \vee x_3) \\ x_3 &= \epsilon \end{aligned}$$

The regular language accepted by the AFA A is obtained by solving the above system of language equations. That is, $x_0 = ab^*a$.

6 Conclusion

Regular expressions have been always centered around NFA and DFA covered in theoretical CS courses. With considerably fewer states than DFA and even NFA, AFA are a generalization of NFA and provide a powerful abstraction layer beyond nondeterminism. Moreover, they are a key ingredient in modeling many software systems, parallel computations, and characterizing model checking algorithms. Language equations have always been far from being well understood in the context of NFA and DFA, but AFA equations, and not the transition diagrams, are in essence the underlying algebraic framework that makes AFA appealing from the point of view of representation and flexibility. However, the systems of equations to be considered involve Boolean expressions over a finite set X of variables and the symbols of an alphabet Σ. The solutions of such systems of equations are precisely the class of regular languages. These language equations may not always be easy to solve algebraically despite being attractive from a theoretical point of view.

References

[1] D. Angluin, S. Eisenstat, and D. Fisman. Learning regular languages via alternating automata. In *Proceedings of the 24th Int. Joint Conference on Artificial Intelligence*, pages 3308–3314. IAAA Press, 2017.

[2] F. Baader and A. Okhotin. On language equations with one-sided concatenation. *Fundamental Informaticae*, 126:1–34, 2013.

[3] S. Berndt, Lutter M. Liskiewicz, M., and R. Reischuk. Learning residual alternating automata. In *Proceedings of the 31st AAAI conference on artificial intelligence*, pages 1749–1755. IAAA Press, 2017.

[4] J. A. Brzozowski and E. L. Leiss. On equations for regular languages, finite automata, and sequential networks. *Theoret. Comput. Sci.*, 10:19–35, 1980.

[5] A.K. Chandra, D.C. Kozen, and L.J. Stockmeyer. Alternation. *J. ACM*, 28:114–133, 1981.

[6] L. D'Antoni, Z. Kincaid, and F. Wang. A symbolic decision procedure for symbolic alternating finite automata. *Electronic Notes in Theoret. Comput. Sci.*, 336:79–99, 2018.

[7] A. Fellah. Equations and regular-like expressions for afa. *Int. J. of Comput. Math*, 51(3-4):157–172, 1994.

[8] A. Fellah. Real-time languages, timed alternating automata, and timed temporal logics: Relationships and specifications. *J. Procedia Computer*, 62:47–54, 2015.

[9] A. Fellah, H. Jürgensen, and S. Yu. Constructions for alternating finite automata. *Int. J. of Comput. Math.*, 35:117–132, 1992.

[10] J.E. Hopcroft, R. Motwani, and Ullman J.D. *Introduction to automata theory, languages, and computation.* Addison-Wesley, Boston, 2001.

[11] J. Kavitha, L. Jeganathan, and G. Sethuraman. Descriptional complexity of alternating finite automta. In *Procedings of the Int. Workshop on Descriptional Complexity of Formal Systems*, pages 188—198, 2016.