Derivative of Regular Expressions with Similarity Relation

BD, FH, MK, DM

December 27, 2025

1 Introduction

Regular expressions and finite automata are two *dual* ways to describe the same regular languages, the former is *bottom-up*, while the latter is *top-down*. In practice, we often start from a regular expression and want to build a deterministic finite automaton (DFA) that recognizes the same language. There are many known constructions for this task, such as Thompson's ε-NFA [1], various forms of position automata [2, 3], or the standard subset construction.

In this work we focus on a different tool for this purpose: *derivatives* of languages and regular expressions [4]. The basic idea of a derivative is simple. When we read a symbol (or a word) from the input, we remove this prefix from all words in the language and look at what is left. The derivative of a language with respect to a word thus describes "what remains" after consuming that word. Each derivative is again a language, and for regular languages it is again regular [4].

A derivative-based DFA construction treats each (equivalence class of) derivative as a state and uses derivatives to define transitions and accepting states. Our main correctness theorem shows that the DFA constructed in this way from a regular expression r accepts exactly the language $\mathcal{L}(r)$: for every word w, the DFA accepts w if and only if $w \in \mathcal{L}(r)$. Moreover, the set of distinct derivative languages of r is finite, and the construction introduces one state for each of them. It is known that no DFA that recognizes $\mathcal{L}(r)$ can have fewer states than the number of such distinct derivatives, so the derivative automaton is minimal [4].

Compared with more classical constructions, the derivative approach has several conceptual advantages. Thompson's construction first builds an NFA with ε-transitions and then applies the subset construction, while position automata mark each symbol occurrence and require a separate follow-set computation [1, 2, 3]. These methods introduce additional intermediate structure such as ε-moves or explicit powersets of NFA states. By contrast, the derivative construction works directly on the syntax of the regular expression and does not need ε-moves or an explicit powerset construction. Each state is a canonical "view" of the same

language after reading some prefix, and the associated DFA is both canonical and minimal for the language [4].

Derivatives are used in this thesis in both the classical (crisp) and fuzzy settings. In the crisp case, we work with the usual regular expressions built from union, concatenation, Kleene star, intersection, and complement, and we use derivatives to construct DFAs that recognize exactly the same languages. We define derivatives semantically (on languages) and syntactically (on regular expressions), prove their semantic-syntactic correspondence, and show that only finitely many derivative languages arise. This finiteness guaranties that the derivative-based construction always terminates and yields minimal DFA.

In the fuzzy case, we add a similarity relation on the alphabet, so that symbols can match not only when they are equal, but also when they are "similar enough". Then we define fuzzy derivatives that take this similarity into account and show that they still lead to regular languages and automata.

Synopsis.

- Section 2 fixes basic notation for alphabets, words, languages, and regular expressions, and recalls their semantics, nullability, and canonical forms.
- Section 3 defines derivatives of regular languages and regular expressions, extends them to words, and establishes their regularity and finiteness properties.
- Section 4 presents the derivative-based DFA construction, specifies its states, transitions, and acceptance condition, and gives pseudocode with a termination argument.
- Section 5 introduces fuzzy derivatives based on cut similarity, extends them to words and regular expressions, and proves regularity and semantic-syntactic correspondence.

2 Preliminaries

We start by providing the foundational definitions necessary for understanding the rest of the paper.

Definition 2.1 (Alphabet, words, and languages)**.**

- *An* alphabet Σ *is a finite nonempty set of symbols.*
- *A* word *over* Σ *is a finite sequence of symbols from* Σ*.*
- *The* empty word *is denoted by* ε*.*
- *The* set of all words *over* Σ *is* Σ^**.*
- *A* language *over* Σ *is a subset* $\mathcal{L} \subseteq \Sigma^*$*.*

Definition 2.2. *Regular expressions over an alphabet Σ are formally defined by the following grammar:*

$$r ::= \emptyset \mid \varepsilon \mid a\,(a \in \Sigma) \mid r + r \mid r \cdot r \mid r^* \mid r\&r \mid \neg r$$

where r ranges over regular expressions and a ranges over symbols in Σ. The intuitive meanings of these expressions are:

- *$\emptyset$ represents the empty language,*
- *ε represents the language containing only the empty word,*
- *a (for $a \in \Sigma$) represents the language containing only the one-letter word a,*
- *$r + s$ represents the union of the languages represented by r and s,*
- *$r \cdot s$ represents the concatenation of languages represented by r and s,*
- *r^* represents the Kleene star of the language represented by r,*
- *$r\&s$ represents the intersection of languages represented by r and s,*
- *$\neg r$ represents the complement of the language represented by r.*

Definition 2.3 (Semantic interpretation)**.** *For a regular expression r over Σ, its semantic interpretation is the language $\mathcal{L}(r) \subseteq \Sigma^*$ defined inductively by:*

$$\begin{aligned}
\mathcal{L}(\emptyset) &= \emptyset, \\
\mathcal{L}(\varepsilon) &= \{\varepsilon\}, \\
\mathcal{L}(a) &= \{a\}, \quad a \in \Sigma, \\
\mathcal{L}(r + s) &= \mathcal{L}(r) \cup \mathcal{L}(s), \\
\mathcal{L}(r \cdot s) &= \{uv \mid u \in \mathcal{L}(r),\ v \in \mathcal{L}(s)\}, \\
\mathcal{L}(r^*) &= \bigcup_{n \geq 0} \mathcal{L}(r)^n, \quad \text{where } \mathcal{L}(r)^0 = \{\varepsilon\},\ \mathcal{L}(r)^{n+1} = \mathcal{L}(r)^n \cdot \mathcal{L}(r), \\
\mathcal{L}(r\&s) &= \mathcal{L}(r) \cap \mathcal{L}(s), \\
\mathcal{L}(\neg r) &= \Sigma^* \setminus \mathcal{L}(r).
\end{aligned}$$

Definition 2.4 (Nullable regular expressions)**.** *In many constructions involving regular expressions it is useful to know whether the language described by a given expression contains the empty word ε. A regular expression r is called* nullable *if $\varepsilon \in \mathcal{L}(r)$.*

We define the nullable *function ν by*

$$\nu(r) = \begin{cases} \varepsilon & \text{if } r \text{ is nullable,} \\ \emptyset & \text{otherwise.} \end{cases}$$

Equivalently, ν is characterized by the following equations:

$$\nu(\varepsilon) = \varepsilon$$
$$\nu(a) = \emptyset \quad (a \in \Sigma)$$
$$\nu(\emptyset) = \emptyset$$
$$\nu(r \cdot s) = \nu(r) \,\&\, \nu(s)$$
$$\nu(r + s) = \nu(r) + \nu(s)$$
$$\nu(r^*) = \varepsilon$$
$$\nu(r \,\&\, s) = \nu(r) \,\&\, \nu(s)$$
$$\nu(\neg r) = \begin{cases} \varepsilon & \text{if } \nu(r) = \emptyset, \\ \emptyset & \text{if } \nu(r) = \varepsilon. \end{cases}$$

Definition 2.5 (Canonical form)**.** *$[r]$ denotes the canonical (normalized) form of the regular expression r. If $[r] = [s]$ for regular expressions r and s, then*

$$\mathcal{L}(r) = \mathcal{L}(s).$$

3 Derivative of Regular Languages and Regular Expressions

The *derivative* of a language $L \subseteq \Sigma^*$ with respect to a symbol $a \in \Sigma$ is defined as the set of all suffixes w such that $aw \in L$. In other words, it consists of all words in L that begin with a, with the leading a removed. Words in L that do not start with a are excluded from the derivative.

Definition 3.1. *The derivative of a language $L \subseteq \Sigma^*$ with respect to symbol $a \in \Sigma$ is defined to be:*

$$\partial_a(L) = \{w \in \Sigma^* \mid a \cdot w \in L\}$$

Example: *Let $\Sigma = \{a, b\}$ and $L = \{aab, bb, baa, a\}$. Then:*

$$\partial_a(L) = \{ab, \varepsilon\}.$$

Since:

- *From aab, removing the initial a yields ab.*
- *From a, removing the initial a yields ε.*
- *The words bb and baa are ignored, since they do not start with a.*

Having defined the derivative of a language with respect to a *single symbol* $a \in \Sigma$, we now extend the concept to handle derivatives with respect to an *entire word* $u \in \Sigma^*$. The idea is to process the word one symbol at a time from left to right: first take the derivative with respect to the initial symbol, and then apply the derivative again to the result with respect to the remaining suffix. This recursive approach leads to the following definition.

Definition 3.2. *The derivative of a language $L \subseteq \Sigma^*$ with respect to a word $u = a \cdot w \in \Sigma^*$ where $a \in \Sigma$ and $w \in \Sigma^*$ is defined to be:*

$$\partial_u(L) = \partial_w(\partial_a(L))$$

Example: *Let $\Sigma = \{a, b\}$ and $L = \{aab, abb, ab, b\}$. Take $u = ab$, so $a \in \Sigma$ and $v = b$.*

First, compute:

$$\partial_a(L) = \{ab, bb, b\},$$

- *$aab \to ab$*
- *$abb \to bb$*
- *$ab \to b$*
- *b is ignored (does not start with a)*

Then:

$$\partial_b\big(\partial_a(L)\big) = \{b, \varepsilon\},$$

because:

- *ab is ignored (does not start with b)*
- *$bb \to b$*
- *$b \to \varepsilon$*

Therefore:

$$\partial_{ab}(L) = \{b, \varepsilon\}.$$

An important property of derivatives is that they preserve regularity: if we start with a regular language, then taking its derivative with respect to any symbol or word ayields another regular language. This fact ensures that derivatives can be used in algorithms and constructions (such as building deterministic finite automata) without leaving the class of regular languages. The following theorem formalizes this property.

Theorem 3.1.

If $L \subseteq \Sigma^*$ is regular then $\partial_u(L)$ is also regular for any word $u \in \Sigma^*$

Proof. Suppose $L \subseteq \Sigma^*$ is a regular language. By definition, there exists a regular expression r such that $\mathcal{L}(r) = L$.

Now consider the syntactic derivative $\mathcal{D}_u(r)$ of r with respect to a word $u \in \Sigma^*$. By Theorem 3.3 we have

$$\mathcal{L}\big(\mathcal{D}_u(r)\big) \;=\; \partial_u(\mathcal{L}(r)) \;=\; \partial_u(L).$$

Furthermore, by construction the derivative $\mathcal{D}_u(r)$ is itself a regular expression, since the rules for derivatives always yield expressions formed from the grammar of regular expressions. Therefore $\partial_u(L)$ is denoted by a regular expression, and hence it is a regular language.

Thus, if $L \subseteq \Sigma^*$ is regular, then $\partial_u(L)$ is regular for every word $u \in \Sigma^*$. $\square$

Theorem 3.2. *Let Σ be a finite alphabet and let $L \subseteq \Sigma^*$ be a regular language. Then the set of all derivatives of L with respect to words over Σ,*

$$\{\, \partial_u(L) \;\mid\; u \in \Sigma^* \},$$

is finite.

Proof. Since L is regular, there exists a regular expression r with $\mathcal{L}(r) = L$. By Theorem 3.3, for every $u \in \Sigma^*$ we have

$$\partial_u(L) \;=\; \partial_u\big(\mathcal{L}(r)\big) \;=\; \mathcal{L}\big(\mathcal{D}_u(r)\big).$$

Thus it is enough to prove that the set

$$\{\, \mathcal{L}(\mathcal{D}_u(r)) \;\mid\; u \in \Sigma^* \}$$

is finite. We prove this by induction on the number N of regular operators in r.

Bases ($N = 0$).

$r = \emptyset$**.**

$$\mathcal{D}_u(\emptyset) = \emptyset \quad \forall\, u \in \Sigma^*.$$

The collection is $\{\emptyset\}$.

$r = \varepsilon$**.**

$$\mathcal{D}_\varepsilon(\varepsilon) = \varepsilon, \qquad \mathcal{D}_a(\varepsilon) = \emptyset \quad \forall\, a \in \Sigma.$$

The collection is $\{\{\varepsilon\}, \emptyset\}$.

$r = a \in \Sigma$**.**

$$\mathcal{D}_\varepsilon(a) = a, \qquad \mathcal{D}_a(a) = \varepsilon, \qquad \mathcal{D}_b(a) = \emptyset \quad \forall\, b \in \Sigma \setminus \{a\}.$$

The collection is finite.

Inductive step ($N > 0$). Induction hypothesis: every regular expression with at most N regular operators has only finitely many languages of the form $\{\, \mathcal{L}(\mathcal{D}_u(\cdot)) \mid u \in \Sigma^* \}$. Let r be a regular expression with $N+1$ regular operators, and let p and q be its subexpressions with at most N regular operators.

(1) Boolean combinations. If $r = p + q$, then for every word u,

$$\mathcal{L}(\mathcal{D}_u(r)) \;=\; \mathcal{L}(\mathcal{D}_u(p)) \cup \mathcal{L}(\mathcal{D}_u(q)).$$

By the induction hypothesis there are only finitely many possibilities for each term on the right, so there are only finitely many possible unions. The same reasoning applies to $\cap$ and $\neg$.

(2) Concatenation. Let $r = p \cdot q$.

For $a \in \Sigma$,

$$\mathcal{D}_a(p \cdot q) \;=\; \mathcal{D}_a(p) \cdot q \;+\; \nu(p) \cdot \mathcal{D}_a(q).$$

For $u = a_1 a_2$,

$$\begin{aligned}\mathcal{D}_{a_1 a_2}(p \cdot q) &= \mathcal{D}_{a_2}\big(\mathcal{D}_{a_1}(p) \cdot q \;+\; \nu(p) \cdot \mathcal{D}_{a_1}(q)\big) \\ &= \mathcal{D}_{a_1 a_2}(p) \cdot q \;+\; \nu\big(\mathcal{D}_{a_1}(p)\big) \cdot \mathcal{D}_{a_2}(q) \;+\; \nu(p) \cdot \mathcal{D}_{a_1 a_2}(q).\end{aligned}$$

For $u = a_1 \cdots a_n$,

$$\mathcal{D}_u(p \cdot q) \;=\; \mathcal{D}_u(p) \cdot q + \nu(p) \cdot \mathcal{D}_u(q) + \nu\big(\mathcal{D}_{a_1}(p)\big) \cdot \mathcal{D}_{a_2 \cdots a_n}(q) + \cdots + \nu\big(\mathcal{D}_{a_1 \cdots a_{n-1}}(p)\big) \cdot \mathcal{D}_{a_n}(q).$$

Equivalently,

$$\mathcal{D}_u(p \cdot q) \;=\; \mathcal{D}_u(p) \cdot q \;+\; \sum_{i=0}^{n-1} \Big(\nu\big(\mathcal{D}_{a_1 \cdots a_i}(p)\big) \cdot \mathcal{D}_{a_{i+1} \cdots a_n}(q)\Big),$$

where the summation denotes a finite union using $+$. By the induction hypothesis, the set of possible terms is finite, so only finitely many unions can occur.

(3) Kleene star. Let $r = p^*$.

For $a \in \Sigma$,

$$\mathcal{D}_a(p^*) \;=\; \mathcal{D}_a(p) \cdot p^*.$$

For $u = a_1 a_2$,

$$\begin{aligned}\mathcal{D}_{a_1 a_2}(p^*) &= \mathcal{D}_{a_2}\big(\mathcal{D}_{a_1}(p) \cdot p^*\big) \\ &= \big(\mathcal{D}_{a_2}\mathcal{D}_{a_1}(p)\big) \cdot p^* \;+\; \nu\big(\mathcal{D}_{a_1}(p)\big) \cdot \mathcal{D}_{a_2}(p^*) \\ &= \mathcal{D}_{a_1 a_2}(p) \cdot p^* \;+\; \nu\big(\mathcal{D}_{a_1}(p)\big) \cdot \big(\mathcal{D}_{a_2}(p) \cdot p^*\big).\end{aligned}$$

Thus $\mathcal{D}_{a_1 a_2}(p^*)$ is a union of terms $(\mathcal{D}_t p) \cdot p^*$.

For $u = a_1 \cdots a_n$ with $n \geq 1$ we prove by induction on n that $\mathcal{D}_u(p^*)$ is a finite union of terms $(\mathcal{D}_t(p)) \cdot p^*$ with $t \neq \varepsilon$. The case $n = 1$ is $\mathcal{D}_a(p^*) = \mathcal{D}_a(p) \cdot p^*$. Inductive step

$$\mathcal{D}_{ua}(p^*) = \mathcal{D}_a\big(\mathcal{D}_u(p^*)\big).$$

By the induction hypothesis, $\mathcal{D}_u(p^*) = \sum_j (\mathcal{D}_{t_j} p) \cdot p^*$. Applying the product rule to each summand and $\mathcal{D}_a(p^*) = \mathcal{D}_a(p) \cdot p^*$ gives

$$\mathcal{D}_a\big((\mathcal{D}_{t_j}(p)) \cdot p^*\big) = (\mathcal{D}_a(\mathcal{D}_{t_j} p)) \cdot p^* \;+\; \nu(\mathcal{D}_{t_j}(p))\,(\mathcal{D}_a(p)) \cdot p^*,$$

which is again a union of terms $(\mathcal{D}_t(p)) \cdot p^*$.

This completes the induction. Therefore the set $\{\mathcal{L}(\mathcal{D}_u(r)) \mid u \in \Sigma^*\}$ is finite, and so $\{\partial_u(L) \mid u \in \Sigma^*\}$ is finite as well. □

So far, we have defined derivatives at the *semantic* level, that is, as operations on languages themselves. However, in practice we often work with the syntactic representation of a language, namely a regular expression. To apply derivatives directly to regular expressions, we need a corresponding *syntactic* definition. This allows us to compute derivatives by manipulating the structure of the expression, rather than first interpreting it as a language. Although the semantic and syntactic derivatives capture the same underlying idea — removing a given prefix from all words in the language — they operate on different objects. The following definition formalizes the derivative in the syntactic setting.

Definition 3.3. *Let R be a regular expression over the alphabet Σ and let $u \in \Sigma^*$ be a word. The* derivative of the regular expression R with respect to u *is the regular expression*

$$\mathcal{D}_u(R)$$

whose language satisfies

$$\mathcal{L}\big(\mathcal{D}_u(R)\big) \;=\; \{\, v \in \Sigma^* \mid u \cdot v \in \mathcal{L}(R)\}.$$

Since we have now defined the meaning of nullable and have a general understanding of derivative we can write the following:

$$\mathcal{D}_a(\varepsilon) = \emptyset$$
$$\mathcal{D}_a(a) = \varepsilon$$
$$\mathcal{D}_a(b) = \emptyset \text{ for } a \neq b$$
$$\mathcal{D}_a(\emptyset) = \emptyset$$
$$\mathcal{D}_a(r \cdot s) = \mathcal{D}_a(r) \cdot s + \nu(r) \cdot \mathcal{D}_a(s)$$
$$\mathcal{D}_a(r^*) = \mathcal{D}_a(r) \cdot r^*$$
$$\mathcal{D}_a(r + s) = \mathcal{D}_a(r) + \mathcal{D}_a(s)$$
$$\mathcal{D}_a(r \,\&\, s) = \mathcal{D}_a(r) \,\&\, \mathcal{D}_a(s)$$
$$\mathcal{D}_a(\neg r) = \neg(\mathcal{D}_a(r))$$

Additional rules may need to be added for special cases, one is for derivative with respect to empty string and the other is for derivative with respect to multiple regular expressions:

$$\mathcal{D}_\varepsilon(r) = r$$
$$\mathcal{D}_{ua}(r) = \mathcal{D}_a(\mathcal{D}_u(r))$$

Examples:

- $\partial_a(a + b) = \partial_a(a) + \partial_a(b) = \varepsilon + \emptyset = \varepsilon$
- $\partial_{aaa}(a^*) = \partial_{aa}(\partial_a(a^*)) = \partial_{aa}(\partial_a(a) \cdot a^*) = \partial_{aa}(\varepsilon \cdot a^*) = \partial_{aa}(a^*) = \cdots = a^*$
- $\partial_c\left((a + b)^* \cdot c\right) = \partial_c((a+b)^*) \cdot c + \nu((a+b)^*) \cdot \partial_c(c) = \emptyset \cdot c + \varepsilon \cdot \varepsilon = \emptyset + \varepsilon = \varepsilon$

We have introduced two notions of derivative: (i) the *semantic derivative* $\partial_u(\mathcal{L})$ of a regular language $\mathcal{L} \subseteq \Sigma^*$ (Definition 3.1), and (ii) the *syntactic derivative* $\mathcal{D}_u(R)$ of a regular expression R (Definition 3.3). Both operations embody the same intuition—"what remains of a word after removing the prefix u"—yet they are defined on different objects. To make their relationship explicit and eliminate any possible ambiguity, we next state a lemma that shows the two definitions coincide under the semantics mapping.

Theorem 3.3 (Semantic–syntactic correspondence)**.** *Let R be a regular expression over Σ and $u \in \Sigma^*$. Then*

$$\mathcal{L}\big(\mathcal{D}_u(R)\big) \;=\; \partial_u\big(\mathcal{L}(R)\big).$$

In words, taking the derivative at the level of regular expressions and then interpreting the result as a language yields exactly the same set of strings as first interpreting R semantically and then taking the derivative of the resulting language.

Proof. We prove that for all regular expressions R and all words $u \in \Sigma^*$,

$$\mathcal{L}\big(\mathcal{D}_u(R)\big) \;=\; \partial_u\big(\mathcal{L}(R)\big).$$

The proof is in two layers: first for a single symbol $a \in \Sigma$ by structural induction on R, then extend to words u by induction on $|u|$.

Step 1 (single symbol). Fix $a \in \Sigma$. We show by structural induction on R that $\mathcal{L}(\mathcal{D}_a(R)) = \partial_a(\mathcal{L}(R))$.
Bases.

- $R = \emptyset$: $\mathcal{D}_a(\emptyset) = \emptyset$ and $\partial_a(\emptyset) = \emptyset$.
- $R = \varepsilon$: $\mathcal{D}_a(\varepsilon) = \emptyset$ and $\partial_a(\{\varepsilon\}) = \emptyset$ since $aw \neq \varepsilon$.
- $R = b \in \Sigma$: $\mathcal{D}_a(b) = \varepsilon$ iff $a = b$, else $\emptyset$; likewise $\partial_a(\{b\}) = \{\varepsilon\}$ iff $a = b$, else $\emptyset$.

Inductive cases. Write $L(\cdot)$ for $\mathcal{L}(\cdot)$.

- $R = R_1 + R_2$:

  $$L(\mathcal{D}_a(R_1{+}R_2)) = L(\mathcal{D}_a(R_1)) \cup L(\mathcal{D}_a(R_2)) = \partial_a L(R_1) \cup \partial_a L(R_2) = \partial_a\big(L(R_1) \cup L(R_2)\big) = \partial_a L(R).$$

- $R = R_1 \,\&\, R_2$:

  $$L(\mathcal{D}_a(R_1 \,\&\, R_2)) = L(\mathcal{D}_a(R_1)) \cap L(\mathcal{D}_a(R_2)) = \partial_a L(R_1) \cap \partial_a L(R_2) = \partial_a\big(L(R_1) \cap L(R_2)\big) = \partial_a L(R).$$

- $R = \neg R_1$:

  $$L(\mathcal{D}_a(\neg R_1)) = \Sigma^* \backslash L(\mathcal{D}_a(R_1)) = \Sigma^* \backslash \partial_a L(R_1) = \partial_a\big(\Sigma^* \backslash L(R_1)\big) = \partial_a L(R).$$

- $R = R_1 \cdot R_2$. Using the derivative rule

  $$\mathcal{D}_a(R_1 \cdot R_2) \;=\; \mathcal{D}_a(R_1) \cdot R_2 \;+\; \nu(R_1) \cdot \mathcal{D}_a(R_2)$$

  and semantics of $+$, $\cdot$, we get

  $$\begin{aligned} \mathcal{L}\big(\mathcal{D}_a(R_1 \cdot R_2)\big) &= \mathcal{L}\big(\mathcal{D}_a(R_1)\big) \cdot \mathcal{L}(R_2) \;\cup\; \mathcal{L}\big(\nu(R_1)\big) \cdot \mathcal{L}\big(\mathcal{D}_a(R_2)\big) \\ &\overset{\text{IH}}{=} \partial_a \mathcal{L}(R_1) \cdot \mathcal{L}(R_2) \;\cup\; \mathcal{L}\big(\nu(R_1)\big) \cdot \partial_a \mathcal{L}(R_2). \end{aligned}$$

  By definition of ν,

  $$\mathcal{L}\big(\nu(R_1)\big) = \begin{cases} \{\varepsilon\} & \text{if } \varepsilon \in \mathcal{L}(R_1), \\ \emptyset & \text{otherwise,} \end{cases}$$

hence

$$\mathcal{L}\big(\mathcal{D}_a(R_1 \cdot R_2)\big) = \partial_a\mathcal{L}(R_1) \cdot \mathcal{L}(R_2) \ \cup \ \begin{cases} \partial_a\mathcal{L}(R_2) & \text{if } \varepsilon \in \mathcal{L}(R_1), \\ \emptyset & \text{otherwise.} \end{cases}$$

This is exactly the left-quotient law for concatenation,

$$\partial_a\big(\mathcal{L}(R_1) \cdot \mathcal{L}(R_2)\big) = \partial_a\mathcal{L}(R_1) \cdot \mathcal{L}(R_2) \ \cup \ \big(\varepsilon \in \mathcal{L}(R_1) \ ? \ \partial_a\mathcal{L}(R_2) : \emptyset\big),$$

so

$$\mathcal{L}\big(\mathcal{D}_a(R_1 \cdot R_2)\big) = \partial_a\big(\mathcal{L}(R_1) \cdot \mathcal{L}(R_2)\big) = \partial_a\mathcal{L}(R_1 \cdot R_2).$$

- $R = (R_1)^*$: Since $L(R_1^*) = \bigcup_{n\geq 0} L(R_1)^n$ and any factorization of a word in $L(R_1^*)$ can be taken with the first block from $L(R_1)$,

  $$\partial_a L(R_1^*) = \partial_a L(R_1) \cdot L(R_1^*).$$

  By the derivative rule $\mathcal{D}_a(R_1^*) = \mathcal{D}_a(R_1) \cdot R_1^*$ and the IH,

  $$L(\mathcal{D}_a(R_1^*)) = L(\mathcal{D}_a(R_1)) \cdot L(R_1^*) = \partial_a L(R_1) \cdot L(R_1^*) = \partial_a L(R_1^*).$$

Thus $L(\mathcal{D}_a(R)) = \partial_a L(R)$ for all a and R.

Step 2 (arbitrary word). We prove by induction on $u \in \Sigma^*$ that $L(\mathcal{D}_u(R)) = \partial_u L(R)$.

- Base $u = \varepsilon$: $\mathcal{D}_\varepsilon(R) = R$ and $\partial_\varepsilon L(R) = L(R)$.
- Step $u = va$:

$$\begin{aligned} L(\mathcal{D}_{va}(R)) &= L\big(\mathcal{D}_a(\mathcal{D}_v(R))\big) = \partial_a\big(L(\mathcal{D}_v(R))\big) \quad \text{(Step 1)} \\ &= \partial_a\big(\partial_v L(R)\big) \quad \text{(IH on } v) = \partial_{va} L(R). \end{aligned}$$

Therefore $L(\mathcal{D}_u(R)) = \partial_u L(R)$ for all words u, completing the proof. □

Theorem 3.4. *Let Σ be a finite alphabet and let R be a regular expression over Σ. Then the set of derivative languages of R,*

$$\big\{\, \mathcal{L}(\mathcal{D}_w(R)) \ \big|\ w \in \Sigma^* \big\}$$

is finite.

Proof. Follows directly from Theorem 3.3 and Theorem 3.2. □

4 Automata Construction

In this section we construct a finite deterministic automaton (DFA) from a regular expression r, using the derivative operator $\mathcal{D}_a(\cdot)$ and the nullable function $\nu(\cdot)$. Each DFA state denotes the remaining language of r after consuming the prefix. Transitions are computed by taking one-step derivatives with respect to input symbols.

4.1 DFA Structure

Let $\equiv$ denote semantic equivalence of regular expressions:

$$R \equiv S \iff \mathcal{L}(R) = \mathcal{L}(S).$$

We define the DFA $M = \langle Q, \Sigma, \delta, q_0, F \rangle$, as follows.

- $Q = \{ [\mathcal{D}_w(r)] \mid w \in \Sigma^* \}$ is the set of states;
- Σ is a finite alphabet.
- $\delta : Q \times \Sigma \to Q$ is the transition function defined by

$$\delta\big([R], a\big) = \big[\mathcal{D}_a(R)\big].$$

- $q_0 = [r] \in Q$ is the start state (since $\mathcal{D}_\varepsilon(r) = r$).
- $F \subseteq Q$ is the set of accepting states:

$$F = \{ [R] \in Q \mid \nu(R) = \varepsilon \}.$$

By Theorem 3.3, $\mathcal{L}(\mathcal{D}_u(r)) = \partial_u(\mathcal{L}(r))$, running the DFA corresponds exactly to repeatedly taking derivatives.

4.2 Pseudocode

- `Sigma` $\equiv \Sigma$.
- `delta` $\equiv \delta : Q \times \Sigma \to Q$.
- `State` $:= [R]$.
- `D(a, R)` $\equiv \mathcal{D}_a(R)$.
- `nullable(R)` $\equiv \nu(R) = \varepsilon$.
- `epsilon` $\equiv \varepsilon$.
- `q_0` $\equiv [r]$ (since $\mathcal{D}_\varepsilon(r) = r$).
- `empty` $\equiv \emptyset$.
- `union`(Q,{x}) $\equiv Q \cup \{x\}$.

```
1  goto(q, a, Q, delta):
2     q_a := [ D(a, q) ]
3     if exists q' in Q and q' = q_a then
4         delta[(q, a)] := q'
5         return (Q, delta)
6     else
```

```
        Q     := union(Q, { q_a })
        delta := union(delta, (q, a) -> q_a))
        for each a' in Sigma do
            (Q, delta) := goto(q_a, a', Q, delta)
        return (Q, delta)

dfa(r):
    q_0 := [ r ]
    Q  := { q_0 }
    delta := empty
    for each a in Sigma do
        (Q, delta) := goto(q_0, a, Q, delta)
    F := { q in Q | nullable(q) }
    return (Q, Sigma, delta, q_0, F)
```

Termination. By Theorem 3.4, only finitely many distinct derivative languages arise; so the construction terminates.

Theorem 4.1. *Let r be a regular expression over Σ, and let $M = \langle Q, \Sigma, \delta, q_0, F\rangle$ be the DFA constructed from r. Then M and r define the same language:*

$$\mathcal{L}(M) \;=\; \mathcal{L}(r).$$

Equivalently, for every word $w \in \Sigma^*$,

$$M \text{ accepts } w \iff w \in \mathcal{L}(r).$$

Proof. Let $\delta^* : Q \times \Sigma^* \to Q$ be the extension of δ to words:

$$\delta^*(q, \varepsilon) = q, \qquad \delta^*(q, wa) = \delta\big(\delta^*(q, w), a\big) \text{ for } w \in \Sigma^*, a \in \Sigma.$$

We first show by induction on $w \in \Sigma^*$ that

$$\delta^*(q_0, w) = \big[\, \mathcal{D}_w(r) \,\big].$$

Base case: For $w = \varepsilon$ we have $q_0 = [r] = [\mathcal{D}_\varepsilon(r)]$, so the claim holds.

Inductive step: Let $w = ua$ with $u \in \Sigma^*$ and $a \in \Sigma$. By the induction hypothesis, $\delta^*(q_0, u) = [\mathcal{D}_u(r)]$. Then, by the definition of δ,

$$\delta^*(q_0, ua) = \delta\big(\delta^*(q_0, u), a\big) = \delta\big([\mathcal{D}_u(r)], a\big) = \big[\, \mathcal{D}_a(\mathcal{D}_u(r)) \,\big] = \big[\, \mathcal{D}_{ua}(r) \,\big],$$

using the rule $\mathcal{D}_{ua}(r) = \mathcal{D}_a(\mathcal{D}_u(r))$. This proves the claim.

Now let $w \in \Sigma^*$. Then:

$$\begin{aligned}
w \text{ is accepted by } M &\iff \delta^*(q_0, w) \in F \\
&\iff \delta^*(q_0, w) = [\mathcal{D}_w(r)] \text{ and } [\mathcal{D}_w(r)] \in F \\
&\iff \nu\big(\mathcal{D}_w(r)\big) = \varepsilon \\
&\iff \varepsilon \in \mathcal{L}\big(\mathcal{D}_w(r)\big) \\
&\iff \varepsilon \in \partial_w\big(\mathcal{L}(r)\big) \quad \text{(by Theorem 3.3)} \\
&\iff w\varepsilon \in \mathcal{L}(r) \quad \text{(by the definition of } \partial_w) \\
&\iff w \in \mathcal{L}(r) \quad \text{(since } w\varepsilon = w).
\end{aligned}$$

Hence $\mathcal{L}(M) = \mathcal{L}(r)$.

□

5 Derivative of Regular Expressions with Fuzzy Similarity Relation

In this paper, we generalize this notion using fuzzy similarity. Suppose $\mathcal{R}$ is a fuzzy similarity relation on Σ, and let μ be a cut value $(0 < \mu \leq 1)$. Then the μ-*fuzzy derivative* of $\mathcal{L}$ with respect to a symbol $a \in \Sigma$, denoted $D_a^\mu(\mathcal{L})$, is defined as the set of suffixes w such that there exists $b \in \Sigma$ with $\mathcal{R}(a, b) \geq \mu$ and $bw \in \mathcal{L}$. In this way, the derivative is taken with respect to all symbols b sufficiently similar to a, with the leading b removed.

Definition 5.1.

$$\partial_a^\mu(\mathcal{L}) = \{w \in \Sigma^* \mid \exists b \in \Sigma \text{ such that } \mathcal{R}(a, b) \geq \mu \text{ and } bw \in \mathcal{L}\}.$$

Alternatively:

$$\partial_a^\mu(\mathcal{L}) = \{w \in \Sigma^* \mid \exists b \in \Sigma \text{ such that } (a, b) \in \mathcal{R}_\mu \text{ and } bw \in \mathcal{L}\}$$

Just as in the classical setting, the notion of a fuzzy derivative with respect to a single symbol can be extended to handle derivatives with respect to an entire word. The idea remains the same: process the word one symbol at a time from left to right, taking the fuzzy derivative with respect to the first symbol and then applying the fuzzy derivative again to the result for the remaining suffix. This recursive approach naturally generalizes the single-symbol fuzzy derivative to arbitrary words, as formalized below.

Definition 5.2. *The fuzzy derivative of a language $\mathcal{L} \subseteq \Sigma^*$ with respect to a word $u = a \cdot v \in \Sigma^*$ where $a \in \Sigma$ and $v \in \Sigma^*$ is defined to be:*

$$\partial_u^\mu(\mathcal{L}) = \partial_v^\mu(\partial_a^\mu(\mathcal{L}))$$

An important property of the classical derivative is that it preserves regularity. This ensures that repeated differentiation never produces a language outside the regular class, making the operation suitable for automata-based algorithms. The same closure property holds in the fuzzy setting: even when derivatives are taken with respect to a cut-based similarity relation, the resulting language remains regular. The following theorem states this formally.

Theorem 5.1. *If $\mathcal{L} \subseteq \Sigma^*$ is regular, then $\partial_u^\mu(\mathcal{L})$ is also regular for any word $u \in \Sigma^*$ and for any cut value $\mu \in [0, 1]$.*

Proof. The argument is similar to the proof of Theorem 3.1. Let $\mathcal{L} \subseteq \Sigma^*$ be regular and fix $\mu \in [0, 1]$.

For a single symbol $a \in \Sigma$ we have, by Definition of ∂_a^μ,

$$\partial_a^\mu(\mathcal{L}) = \{w \in \Sigma^* \mid \exists b \in \Sigma \text{ with } \mathcal{R}(a,b) \geq \mu \text{ and } bw \in \mathcal{L}\} = \bigcup_{\substack{b \in \Sigma \\ \mathcal{R}(a,b) \geq \mu}} \partial_b(\mathcal{L}),$$

where $\partial_b(\mathcal{L})$ is the classical derivative of $\mathcal{L}$ with respect to b. By Theorem 3.1, each $\partial_b(\mathcal{L})$ is regular, and since Σ is finite, the union is finite; hence $\partial_a^\mu(\mathcal{L})$ is regular.

For a word $u = a_1 \cdots a_n$ we use the recursive definition $\partial_u^\mu(\mathcal{L}) = \partial_{a_n}^\mu(\cdots \partial_{a_1}^\mu(\mathcal{L}) \cdots)$ and apply the previous argument inductively: at each step we take a fuzzy derivative of a regular language and obtain a regular language again. Thus $\partial_u^\mu(\mathcal{L})$ is regular for every $u \in \Sigma^*$ and every cut $\mu \in [0,1]$. □

In order to compute fuzzy derivatives in practice, it is convenient to identify, for a given symbol and cut value, the set of symbols that are considered similar enough to be treated as matches. We call this set the *cut neighborhood* of the symbol, and define it formally as follows.

Definition 5.3. *For $a \in \Sigma$ and $\mu \in [0:1]$ we define cut neighborhood as*

$$\mathcal{S}_a^\mu = \{b \in \Sigma : (a,b) \in \mathcal{R}_\mu\}$$

As in the classical case, we distinguish between the *semantic* fuzzy derivative, defined directly on languages, and the *syntactic* fuzzy derivative, defined on regular expressions. The syntactic formulation allows us to compute derivatives by manipulating the structure of the expression itself, without first converting it to its language. In the fuzzy setting, this requires taking into account all symbols in the cut neighborhood of the given symbol. The following definition formalizes the syntactic fuzzy derivative.

Definition 5.4. *Let R be a regular expression over the alphabet Σ, let $a \in \Sigma$ be a symbol and let $\mu \in [0:1]$. The* fuzzy derivative of the regular expression R with respect to a and μ *is the regular expression*

$$\mathcal{D}_a^\mu(R)$$

whose language satisfies

$$\mathcal{L}\big(\mathcal{D}_a^\mu(R)\big) = \{v \in \Sigma^* : \exists b \in \mathcal{S}_a^\mu \wedge b \cdot v \in \mathcal{L}(R)\}$$

We define fuzzy derivative for regular expressions

$$\mathcal{D}_a^\mu(\varepsilon) = \emptyset$$
$$\mathcal{D}_a^\mu(b) = \begin{cases} \varepsilon & \text{if } b \in \mathcal{S}_a^\mu, \\ \emptyset & \text{otherwise.} \end{cases}$$
$$\mathcal{D}_a^\mu(\emptyset) = \emptyset$$
$$\mathcal{D}_a^\mu(r \cdot s) = \mathcal{D}_a^\mu(r) \cdot s + \nu(r) \cdot \mathcal{D}_a^\mu(s)$$
$$\mathcal{D}_a^\mu(r^*) = \mathcal{D}_a^\mu(r) \cdot r^*$$
$$\mathcal{D}_a^\mu(r + s) = \mathcal{D}_a^\mu(r) + \mathcal{D}_a^\mu(s)$$
$$\mathcal{D}_a^\mu(r \,\&\, s) = \mathcal{D}_a^\mu(r) \,\&\, \mathcal{D}_a^\mu(s)$$
$$\mathcal{D}_a^\mu(\neg r) = \neg(\mathcal{D}_a^\mu(r))$$

The semantic and syntactic definitions of the fuzzy derivative are intended to capture the same intuitive operation: removing a given prefix (up to similarity) from all words in the language. To confirm this alignment, we state the following correspondence theorem, which ensures that taking the fuzzy derivative at the syntactic level and then interpreting it as a language yields exactly the same result as first interpreting the expression and then applying the semantic fuzzy derivative.

Theorem 5.2 (Semantic-syntactic correspondence)**.** *Let R be a regular expression over Σ, $u \in \Sigma^*$ and $\mu \in [0:1]$. Then*

$$\mathcal{L}\big(\mathcal{D}_u^\mu(R)\big) = \partial_u^\mu\big(\mathcal{L}(R)\big)$$

Proof. We prove that for all regular expressions R and all words $u \in \Sigma^*$,

$$\mathcal{L}\big(\mathcal{D}_u^\mu(R)\big) \;=\; \partial_u^\mu\big(\mathcal{L}(R)\big).$$

As in the crisp case, the proof has two steps: first a single symbol $a \in \Sigma$, then an arbitrary word u.

Step 1 (single symbol). Fix $a \in \Sigma$. We show by structural induction on R that

$$\mathcal{L}\big(\mathcal{D}_a^\mu(R)\big) = \partial_a^\mu\big(\mathcal{L}(R)\big).$$

Bases.

- $R = \emptyset$: $\mathcal{D}_a^\mu(\emptyset) = \emptyset$, hence $\mathcal{L}(\mathcal{D}_a^\mu(\emptyset)) = \emptyset$, and $\partial_a^\mu(\emptyset) = \emptyset$ by definition.
- $R = \varepsilon$: $\mathcal{D}_a^\mu(\varepsilon) = \emptyset$, so $\mathcal{L}(\mathcal{D}_a^\mu(\varepsilon)) = \emptyset$; on the language side there is no word of the form bw (with $(a, b) \in \mathcal{R}_\mu$) equal to ε, hence $\partial_a^\mu(\{\varepsilon\}) = \emptyset$.
- $R = b \in \Sigma$: By Definition 5.4, $\mathcal{D}_a^\mu(b) = \varepsilon$ if $b \in \mathcal{S}_a^\mu$ and $\mathcal{D}_a^\mu(b) = \emptyset$ otherwise, so
$$\mathcal{L}(\mathcal{D}_a^\mu(b)) = \begin{cases} \{\varepsilon\} & \text{if } b \in \mathcal{S}_a^\mu, \\ \emptyset & \text{otherwise.} \end{cases}$$

On the language side we have

$$\partial_a^\mu(\{b\}) = \{w \mid \exists c \in \Sigma,\ (a,c) \in \mathcal{R}_\mu \text{ and } cw = b\},$$

which is $\{\varepsilon\}$ exactly when $(a,b) \in \mathcal{R}_\mu$ (i.e. $b \in \mathcal{S}_a^\mu$), and $\emptyset$ otherwise. Thus the two coincide.

Inductive cases. For $R = R_1 + R_2$, $R = R_1 \,\&\, R_2$, $R = \neg R_1$, $R = R_1 \cdot R_2$ and $R = (R_1)^*$, the syntactic fuzzy derivative is defined by exactly the same algebraic clauses as in the crisp case (the parameter μ does not appear in these rules), and the semantic fuzzy derivative ∂_a^μ is defined as a finite union of classical derivatives:

$$\partial_a^\mu(L) = \bigcup_{\substack{b \in \Sigma \\ (a,b) \in \mathcal{R}_\mu}} \partial_b(L).$$

Since each classical derivative ∂_b satisfies the usual algebraic laws for $+$, $\&$, $\neg$, concatenation, and star, and finite unions preserve these equalities, the calculations in the crisp proof carry over verbatim with $\mathcal{D}_a$ and ∂_a replaced by $\mathcal{D}_a^\mu$ and ∂_a^μ. Using the induction hypothesis on R_1 and R_2 in each case, we obtain

$$\mathcal{L}\big(\mathcal{D}_a^\mu(R)\big) = \partial_a^\mu\big(\mathcal{L}(R)\big)$$

for all $a \in \Sigma$ and all regular expressions R.

Step 2 (arbitrary word). We prove by induction on $u \in \Sigma^*$ that

$$\mathcal{L}\big(\mathcal{D}_u^\mu(R)\big) = \partial_u^\mu\big(\mathcal{L}(R)\big).$$

- Base $u = \varepsilon$: $\mathcal{D}_\varepsilon^\mu(R) = R$ and $\partial_\varepsilon^\mu(\mathcal{L}(R)) = \mathcal{L}(R)$.
- Step $u = va$ with $v \in \Sigma^*$ and $a \in \Sigma$:

$$\begin{aligned}
\mathcal{L}\big(\mathcal{D}_{va}^\mu(R)\big) &= \mathcal{L}\big(\mathcal{D}_a^\mu(\mathcal{D}_v^\mu(R))\big) \\
&= \partial_a^\mu\big(\mathcal{L}(\mathcal{D}_v^\mu(R))\big) \quad \text{(by Step 1)} \\
&= \partial_a^\mu\big(\partial_v^\mu(\mathcal{L}(R))\big) \quad \text{(IH on } v\text{)} \\
&= \partial_{va}^\mu\big(\mathcal{L}(R)\big).
\end{aligned}$$

Therefore $\mathcal{L}(\mathcal{D}_u^\mu(R)) = \partial_u^\mu(\mathcal{L}(R))$ for all words $u \in \Sigma^*$, which completes the proof. □

Examples

Assume the alphabet $\Sigma = \{a, b, c\}$ and a fuzzy similarity relation $\mathcal{R}$ on Σ defined as follows:

$\mathcal{R}$	a	b	c
a	1	0.8	0.4
b	0.8	1	0.5
c	0.4	0.5	1

Let the cut value be $\mu = 0.7$, so the μ-cut of $\mathcal{R}$ includes the pairs:

$$\mathcal{R}_\mu = \{(a,a),(a,b),(b,a),(b,b),(c,c)\}$$

We can also observe that:

$$\mathcal{S}_a^\mu = \{a,b\}$$
$$\mathcal{S}_b^\mu = \{a,b\}$$
$$\mathcal{S}_a^\mu = \{c\}$$

Let $\mathcal{L} = \{abc, ba, bb\}$. Then:

- $\partial_a^\mu(\mathcal{L}) = \{bc, a, b\}$
  Because:
  - $(a,a) \in \mathcal{R}_\mu$ and $abc \in \mathcal{L} \Rightarrow bc \in \partial_a^\mu(\mathcal{L})$
  - $(a,b) \in \mathcal{R}_\mu$ and $ba \in \mathcal{L} \Rightarrow a \in \partial_a^\mu(\mathcal{L})$
  - $(a,b) \in \mathcal{R}_\mu$ and $bb \in \mathcal{L} \Rightarrow b \in \partial_a^\mu(\mathcal{L})$
- $\partial_c^\mu(\mathcal{L}) = \emptyset$
  Because:
  - Only $(c,c) \in \mathcal{R}_\mu$ with $\mathcal{R}(c,c) = 1 \geq \mu$
  - But $cb, ca, cc \notin \mathcal{L} \Rightarrow$ no valid bw with $(c,b) \in \mathcal{R}_\mu$ and $bw \in \mathcal{L}$
- $\partial_b^\mu(\mathcal{L}) = \{bc, a, b\}$
  Because:
  - $(b,a) \in \mathcal{R}_\mu$ and $abc \in \mathcal{L} \Rightarrow bc \in \partial_b^\mu(\mathcal{L})$
  - $(b,b) \in \mathcal{R}_\mu$ and $ba \in \mathcal{L} \Rightarrow a \in \partial_b^\mu(\mathcal{L})$
  - $(b,b) \in \mathcal{R}_\mu$ and $bb \in \mathcal{L} \Rightarrow b \in \partial_b^\mu(\mathcal{L})$

Theorem 5.3. *Let Σ be a finite alphabet and let $\mathcal{L} \subseteq \Sigma^*$ be a regular language. For every cut value $\mu \in [0,1]$, the set of fuzzy derivatives*

$$\left\{ \partial_u^\mu(\mathcal{L}) \;\middle|\; u \in \Sigma^* \right\}$$

is finite.

Proof. Fix $\mu \in [0,1]$ and let $\mathcal{L}$ be regular. By Theorem 3.2, the set of classical derivatives

$$D \;=\; \left\{ \partial_w(\mathcal{L}) \mid w \in \Sigma^* \right\}$$

is finite.

For a single symbol $a \in \Sigma$ we have, by the definition of the fuzzy derivative,

$$\partial_a^\mu(\mathcal{L}) = \{w \mid \exists b \in \Sigma \text{ with } \mathcal{R}(a,b) \geq \mu \text{ and } bw \in \mathcal{L}\} = \bigcup_{\substack{b \in \Sigma \\ \mathcal{R}(a,b) \geq \mu}} \partial_b(\mathcal{L}),$$

so $\partial_a^\mu(\mathcal{L})$ is a finite union of elements of D.

Using the recursive definition $\partial_{av}^\mu(\mathcal{L}) = \partial_v^\mu(\partial_a^\mu(\mathcal{L}))$ and the fact that ∂_a^μ distributes over unions, a straightforward induction on the length of u shows that every fuzzy derivative $\partial_u^\mu(\mathcal{L})$ is a finite union of languages from D. Since D is finite, there are only finitely many such unions (at most $2^{|D|}$). Hence the set $\{\partial_u^\mu(\mathcal{L}) \mid u \in \Sigma^*\}$ is finite. □

Theorem 5.4. *Let Σ be a finite alphabet and let R be a regular expression over Σ. For every cut value $\mu \in [0,1]$, the set of derivative languages*

$$\big\{\, \mathcal{L}(\mathcal{D}_w^\mu(R)) \;\big|\; w \in \Sigma^* \big\}$$

is finite.

Proof. Let $\mu \in [0,1]$ be fixed and $\mathcal{L} = \mathcal{L}(R)$. By Theorem 5.2, for every word $w \in \Sigma^*$,

$$\mathcal{L}\big(\mathcal{D}_w^\mu(R)\big) \;=\; \partial_w^\mu\big(\mathcal{L}(R)\big) \;=\; \partial_w^\mu(\mathcal{L}).$$

Hence

$$\big\{\, \mathcal{L}(\mathcal{D}_w^\mu(R)) \mid w \in \Sigma^* \big\} \;=\; \big\{\, \partial_w^\mu(\mathcal{L}) \mid w \in \Sigma^* \big\},$$

and the right-hand set is finite by Theorem 5.3. This proves the claim. □

Theorem 5.5. *Let r be a regular expression over Σ, let $\mu \in [0,1]$, and let $M^\mu = \langle Q^\mu, \Sigma, \delta^\mu, q_0^\mu, F^\mu \rangle$ be the DFA constructed from r using fuzzy derivatives (for the fixed cut value μ). Then M^μ and r define the same μ-language:*

$$\mathcal{L}(M^\mu) \;=\; \mathcal{L}^\mu(r).$$

Equivalently, for every word $w \in \Sigma^*$,

$$M^\mu \text{ accepts } w \iff w \in \mathcal{L}^\mu(r).$$

Proof. Let $\delta^{\mu *} : Q^\mu \times \Sigma^* \to Q^\mu$ be the standard extension of δ^μ to words:

$$\delta^{\mu *}(q, \varepsilon) = q, \qquad \delta^{\mu *}(q, wa) = \delta^\mu\big(\delta^{\mu *}(q, w), a\big) \text{ for } w \in \Sigma^*, a \in \Sigma.$$

We first show by induction on $w \in \Sigma^*$ that

$$\delta^{\mu *}(q_0^\mu, w) = \big[\, \mathcal{D}_w^\mu(r) \,\big].$$

Base case: For $w = \varepsilon$ we have $q_0^\mu = [r] = [\mathcal{D}_\varepsilon^\mu(r)]$, so the claim holds.

Inductive step: Let $w = ua$ with $u \in \Sigma^*$ and $a \in \Sigma$. By the induction hypothesis, $\delta^{\mu *}(q_0^\mu, u) = [\mathcal{D}_u^\mu(r)]$. Then, by the definition of δ^μ,

$$\delta^{\mu *}(q_0^\mu, ua) = \delta^\mu\big(\delta^{\mu *}(q_0^\mu, u), a\big) = \delta^\mu\big([\mathcal{D}_u^\mu(r)], a\big) = \big[\, \mathcal{D}_a^\mu(\mathcal{D}_u^\mu(r)) \,\big] = \big[\, \mathcal{D}_{ua}^\mu(r) \,\big],$$

using the rule $\mathcal{D}_{ua}^\mu(r) = \mathcal{D}_a^\mu(\mathcal{D}_u^\mu(r))$. This proves the claim.

Now let $w \in \Sigma^*$. Then:

$$
\begin{aligned}
w \text{ is accepted by } M^\mu &\iff \delta^{\mu *}(q_0^\mu, w) \in F^\mu \\
&\iff \delta^{\mu *}(q_0^\mu, w) = [\mathcal{D}_w^\mu(r)] \text{ and } [\mathcal{D}_w^\mu(r)] \in F^\mu \quad \text{(by the claim above and def. of } F^\mu) \\
&\iff \nu\big(\mathcal{D}_w^\mu(r)\big) = \varepsilon \\
&\iff \varepsilon \in \mathcal{L}\big(\mathcal{D}_w^\mu(r)\big) \\
&\iff \varepsilon \in \partial_w^\mu\big(\mathcal{L}(r)\big) \quad \text{(by Theorem 5.2)} \\
&\iff w \in \mathcal{L}^\mu(r) \quad \text{(by the definition of } \mathcal{L}^\mu(r)).
\end{aligned}
$$

Hence $\mathcal{L}(M^\mu) = \mathcal{L}^\mu(r)$, as required. □

References

[1] Ken Thompson. Programming techniques: Regular expression search algorithm. *Commun. ACM*, 11(6):419–422, June 1968.

[2] V. M. Glushkov. The abstract theory of automata. *Russ. Math. Surv.*, 16(5):1–53, 1962.

[3] R. McNaughton and H. Yamada. Regular expressions and state graphs for automata. *IRE Transactions on Electronic Computers*, EC-9(1):39–47, 1960.

[4] J.A. Brzozowski. Canonical regular expressions and minimal state graphs for definite events. *Mathematical Theory of Automata*, 12:529–561, 1962.