Fundamenta Informaticae XX (2016) 1–24
DOI 10.3233/FI-2012-0000
IOS Press

Distinguishability Operations and Closures

Cezar Câmpeanu
Department of Computer Science
The University of Prince Edward Island, Charlottetown, PE, Canada
ccampeanu@upei.ca

Nelma Moreira*, **Rogério Reis***
Centro de Matemática e Faculdade de Ciências da Universidade do Porto,
4169-007 Porto, Portugal
{nam,rvr}@dcc.fc.up.pt

Abstract. Given a language L, we study the language of words $\mathsf{D}(L)$, that distinguish between pairs of different left quotients of L. We characterize this distinguishability operation, show that its iteration has always a fixed point, and we generalize this result to operations derived from closure operators and Boolean operators. For the case of regular languages, we give an upper bound for the state complexity of the distinguishability operation, and prove its tightness. We show that the set of minimal words that can be used to distinguish between different left quotients of a regular language L has at most $n-1$ elements, where n is the state complexity of L, and we also study the properties of its iteration. We generalize the results for the languages of words that distinguish between pairs of different right quotients and two-sided quotients of a language L.

Keywords: Formal Languages, Myhill-Nerode Relations, Quotients, Language Operations, Closures, Regular Languages, State Complexity.

1. Introduction

Regular languages and operations over them have been extensively studied during the last sixty years, the applications of automata being continuously extended in different areas. As a practical example, we

*This work was partially funded by the European Regional Development Fund through the programme COMPETE and by the Portuguese Government through the FCT under projects PEst-C/MAT/UI0144/2013 and FCOMP-01-0124-FEDER-020486.

can use automata to model various electronic circuits. The testing of the circuits can be done by applying several inputs to various pins of a circuit, and checking the output produced. Because in many cases the circuits emulate automata, it is useful to develop general tools for testing various properties of automata, such as testing the relation between the response of the circuit for the same signal, applied to different gates. To answer if an automaton is minimal requires to test if two states are equivalent or not. The easiest way to do this is to use as input different words, and see if for both states, we reach states with the same finality, thus, in case of a circuit, for both input gates we will get the same value of the output bit. However, checking every possible word is a tedious task, and it would be useful to limit the testing only to the words that can distinguish between states. Therefore, it is worth studying the languages that distinguish between all non-equivalent states of a given deterministic finite automaton.

For an automaton $\mathcal{A}$ we denote the distinguishing language by $\mathsf{D}(\mathcal{A})$ and the language of minimal words distinguishing between all non-equivalent states by $\underline{\mathsf{D}}(\mathcal{A})$. We can also consider the distinguishability languages $\mathsf{D}(L)$, and $\underline{\mathsf{D}}(L)$ for a regular language, L, which will distinguish between all non-equivalent words. Moreover, the distinguishability operation can be defined for arbitrary languages.

The idea of studying word or state distinguishability is not new. In 1958, Ginsburg studied the length of the smallest uniform experiment which distinguishes the terminal states of a machine [14], and with Spanier in [15], he studies whether or not an arbitrary semigroup can serve as an input for a machine that distinguishes between the elements of the input semigroup. A comparable work was done for terminal distinguishability by Sempere [22], where terminal segments of automata are studied to characterize language families that can be identified in the limit from positive data. Indeed, knowing that an automaton $\mathcal{A}$ has at most n states, and having the language $\underline{\mathsf{D}}(\mathcal{A})$, together with the words of length at most $n+1$ that are in the language $\mathcal{L}(\mathcal{A})$, we can recover the initial automaton $\mathcal{A}$. Note that without the language $\underline{\mathsf{D}}(\mathcal{A})$, any learning procedure will only approximate the language $\mathcal{L}(\mathcal{A})$. For example, in case we know M to be the set of all the words of a language L with length at most $n+1$, we can infer that L is a cover language for M, but we cannot determine which one of these cover languages is L. Thus, any learning procedure would only be able to guess L from M, and the guess would not be accurate, as the number of cover automata for a finite language can be staggeringly high [8, 11]. In [20], Restivo and Vaglica proposed a graph-theoretical approach to test automata minimality. For a given automaton $\mathcal{A}$ they associate a digraph, called *pair graph*, where vertices are pairs $\{p, q\}$ of states of $\mathcal{A}$, and edges connect vertices for which the states have a transition from the same symbol in $\mathcal{A}$. Then, two states p and q of $\mathcal{A}$ are distinguishable if and only if there is a path from the vertex $\{p, q\}$ to a vertex $\{p', q'\}$, where p' is final and q' is non final, i.e., there exists a word that distinguishes between them. A related research topic is the problem of finding a minimal DFA that distinguishes between two words by accepting one and rejecting the other. It was studied by Blumer et al. in [2], and recently Demaine et al. in [13] reviewed several attempts to solve the problem and presented new results.

In the present paper we do not separate two words by a language; instead, we distinguish between non-equivalent quotients of the same language. In Section 2, we introduce the notations we are going to use. In Section 3, we define and prove general properties of the distinguishability operation on arbitrary languages. For regular languages, we study the state complexity of the distinguishability operation in Section 4. We use many powerful tools such as language quotients, atoms, and universal witnesses that hide proof complexity, helping us to produce a presentation easier to follow. Afterwards, in Section 5, we analyze the set of minimal words with respect to quasi-lexicographical order that distinguishes different quotients of a language and give several characterizations when the language is regular.

In Section 6, we present an algorithm that can be used as a positive learning procedure for a language

L if $\underline{\mathsf{D}}(L)$ is known. In Section 7, we present a class of operands using closure operations and Boolean operations, operands that have a fixed point under iteration. In Section 8, we define other distinguishability operations and study their properties. The conclusions, together with open problems and future work, are included in Section 9. A preliminary version of some of the results presented here has previously appeared in [9].

2. Notation and Definitions

For a set T, its cardinality is denoted by $|T|$. An *alphabet* Σ is a finite non-empty set, and the free monoid generated by Σ is $\Sigma^\star$. A word w is an element of $\Sigma^\star$ and a *language* is a subset of $\Sigma^\star$. The *complement* of a language L is $\overline{L} = \Sigma^\star \setminus L$. The *length* of a word $w \in \Sigma^\star$, $w = a_1 a_2 \dots a_n$, $a_i \in \Sigma$, $1 \leq i \leq n$, with $n \in \mathbb{N}$ is $|w| = n$. The *empty word* is ε, and $|\varepsilon| = 0$. If $w = uxv$ for some $u, v, x \in \Sigma^\star$ then u is a *prefix* of w, x is a *factor* (or *infix*) of w, and v a *suffix* of w. Given an arbitrary language L over an alphabet Σ, the set of suffixes of L is defined by $\mathsf{suff}(L) = \{w \mid xw \in L\}$. In the same way, we define the set of prefixes of L, $\mathsf{pref}(L)$, and the set of infixes of L, $\mathsf{infix}(L)$.

The reverse w^R of a word $w \in \Sigma^\star$ is defined as follows: $\varepsilon^R = \varepsilon$, and $(wa)^R = aw^R$, for $a \in \Sigma$. The *reverse* of a language L is denoted by L^R and defined as $L^R = \{w^R \mid w \in L\}$. Consider an order over Σ. In $\Sigma^\star$, we define the *quasi-lexicographical* order as: $w \preceq w'$ if $|w| < |w'|$ or $|w| = |w'|$ and w lexicographically precedes w'.

A language L induces on $\Sigma^\star$ the Myhill-Nerode equivalence relation: $x \equiv_L y$ if, for all $w \in \Sigma^\star$, we have that $xw \in L$ if and only if $yw \in L$. The *left quotient*, or simply *quotient*, of an arbitrary language L by a word w is the language $w^{-1}L = \{x \mid wx \in L\}$. A quotient corresponds to an equivalence class of $\equiv_L$, i.e., two words are equivalent if and only if their quotients are the same.

A *deterministic finite automaton* (DFA) is a quintuple $\mathcal{A} = \langle Q, \Sigma, q_0, \delta, F\rangle$, where Q is a finite non-empty set, the set of states, Σ is the alphabet, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, and $\delta : Q \times \Sigma \longrightarrow Q$ is the transition function. This function defines for each symbol of the alphabet a transformation of the set Q of states (i.e., a map from Q to Q). The *transition semigroup* of a DFA $\mathcal{A}$, [1], is the semigroup of transformations of Q generated by the transformations induced by the symbols of Σ. A *reduced* DFA is a DFA with all states reachable from the initial state (accessible), and all states can reach a final state (useful), except at most one that is a *sink* state or *dead* state, i.e., a state where all output transitions are self loops. In the case of an incomplete automaton the transition function is a partial function and we can add the dead state to make the automaton complete. A *trimmed* DFA is a DFA with all states reachable and all states useful, thus it may be incomplete, i.e., where not all transitions are defined. The transition function δ can be extended to $\delta : Q \times \Sigma^\star \longrightarrow Q$ by $\delta(q, \varepsilon) = q$, and $\delta(q, wa) = \delta(\delta(q, w), a)$.

The language recognized by a DFA $\mathcal{A}$ is $\mathcal{L}(\mathcal{A}) = \{w \in \Sigma^\star \mid \delta(q_0, w) \in F\}$. We denote by L_q and R_q the *left* and *right* languages of q, respectively, i.e., $L_q = \{w \mid \delta(q_0, w) = q\}$, and $R_q = \{w \mid \delta(q, w) \in F\}$. The minimal word in quasi-lexicographical order that reaches state $q \in Q$ is $x_{\mathcal{A}}(q)$; the word $x_{\mathcal{A}}(q)$ is also the minimal element of L_q. If $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F\rangle$ is a DFA recognizing the language L and $R_q = R_p$, then we say that p and q are equivalent, and write $p \equiv_{\mathcal{A}} q$. A DFA is *minimal* if it has no equivalent states.

A *regular language* is a language recognized by a DFA. If a language L is regular, the number of distinct left quotients is finite, and it is exactly the number of states in the *minimal* DFA recognizing L.

This number is called the *state complexity* of L, and it is denoted by $sc(L)$. In a minimal DFA, for each $q \in Q$, R_q is exactly a quotient. If some quotient of a language L is $\emptyset$, this means that the minimal DFA of L has a dead state.

A *nondeterministic finite automata* (NFA) is a quintuple $\mathcal{N} = \langle Q, \Sigma, I, \delta, F\rangle$, where Q, Σ, and F are the same as in the DFA definition, $I \subseteq Q$ is the set of initial states, and $\delta : Q \times \Sigma \longrightarrow 2^Q$ is the transition function. The transition function can also be extended to subsets of Q instead of states, and to words instead of symbols of Σ. The language recognized by an NFA $\mathcal{N}$ is $\mathcal{L}(\mathcal{N}) = \{w \mid \delta(I, w) \cap F \neq \emptyset\}$. It is obvious that a DFA is also an NFA. Any NFA can be converted into an equivalent DFA by the well-known *subset construction*. Given an NFA $\mathcal{N}$ for L, an NFA $\mathcal{N}^R$ for L^R is obtained by interchanging the sets of final and initial states of $\mathcal{N}$ and reversing all transitions between states.

More notation and definitions related to formal languages can be looked up in [21, 23].

3. The (Left) Distinguishability Operation

Let L be a regular language. For every pair of words, $x, y \in \Sigma^\star$, with $x \not\equiv_L y$, there exists at least one word w such that either $xw \in L$ or $yw \in L$. Let $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F\rangle$ be a DFA such that $L = \mathcal{L}(\mathcal{A})$. If two states $p, q \in Q$, $p \not\equiv_A q$, then there exists at least one word w such that $\delta(p, w) \in F \nLeftrightarrow \delta(q, w) \in F$. We say that w distinguishes between the words x and y, in the first case, and the states p and q, in the second case. Given $x, y \in \Sigma^\star$, the language that *distinguishes* x from y w.r.t. L is

$$\mathsf{D}_L(x, y) = \{w \mid xw \in L \nLeftrightarrow yw \in L\}\,. \tag{1}$$

Naturally, we define the *left distinguishability language* (or simply, distinguishability language) of L by

$$\mathsf{D}(L) = \{w \mid \exists x, y \in \Sigma^\star\ (xw \in L\ \wedge\ yw \notin L)\}. \tag{2}$$

It is immediate that $\mathsf{D}(L) = \bigcup_{x,y\in\Sigma^\star} \mathsf{D}_L(x, y)$.

In the same way, for the DFA $\mathcal{A}$, we define $\mathsf{D}_L(p, q)$ for $p, q \in Q$, and

$$\mathsf{D}(\mathcal{A}) = \{w \mid \exists p, q \in Q\ (\delta(p, w) \in F\ \wedge\ \delta(q, w) \notin F)\}. \tag{3}$$

Lemma 3.1. Let $\mathcal{A}_1, \mathcal{A}_2$ be two reduced DFAs such that $\mathcal{L}(\mathcal{A}_1) = \mathcal{L}(\mathcal{A}_2) = L$. Then $\mathsf{D}(\mathcal{A}_1) = \mathsf{D}(\mathcal{A}_2) = \mathsf{D}(L)$.

Proof:
Let $\mathcal{A}_1 = \langle Q, \Sigma, q_0, \delta, F\rangle$ and $L = \mathcal{L}(\mathcal{A}_1) = \mathcal{L}(\mathcal{A}_2)$. It is enough to prove that $\mathsf{D}(L) = \mathsf{D}(\mathcal{A}_1)$. If $w \in \mathsf{D}(L)$, then we have two words $x, y \in \Sigma^\star$ such that $xw \in L$ and $yw \notin L$. Let $p = \delta(q_0, x)$ and $q = \delta(q_0, y)$. Then, $\delta(q_0, xw) \in F$ and $\delta(q_0, yw) \notin F$, so $w \in \mathsf{D}(\mathcal{A}_1)$. If $w \in \mathsf{D}(\mathcal{A}_1)$, then there exist $p, q \in Q$ such that $\delta(p, w) \in F$ and $\delta(q, w) \notin F$; as $\mathcal{A}_1$ is reduced, there must exist x, y with $\delta(q_0, x) = p$ and $\delta(q_0, y) = q$, therefore $\delta(q_0, xw) = \delta(p, w) \in F$ and $\delta(q_0, yw) = \delta(q, w) \notin F$, hence, $xw \in L$ and $yw \notin L$, i.e., we conclude $w \in \mathsf{D}(L)$. □

This shows that the operator D is independent of the automata we choose to represent the language, as long as both automata are reduced.

The distinguishability operation can be extended to arbitrary languages by using the suffix operator:

$$\mathsf{D}(L) = \mathsf{suff}(L) \cap \mathsf{suff}(\bar{L}). \tag{4}$$

It is easy to see that both definitions (2) and (4) are equivalent. In what follows, we present some characterization results for the distinguishability operation, and show that iterating the operation always leads to a fixed point.

The distinguishability operation can be expressed directly by means of language quotients, as it can be seen in the following result.

Theorem 3.2. Let L be an arbitrary language with $\left\{w^{-1}L \mid w \in \Sigma^\star\right\}$ being its set of (left) quotients. Then we have the equality

$$\mathsf{D}(L) = \bigcup_{x\in\Sigma^\star} x^{-1}L \setminus \bigcap_{x\in\Sigma^\star} x^{-1}L.$$

Proof:
We first consider the cases where L is a trivial language over Σ, i.e., $L = \emptyset$ or $L = \Sigma^\star$.

If $L = \emptyset$, then for any $w \in \Sigma$ we have $w^{-1}L = \emptyset = L$ and $\mathsf{D}(L) = \emptyset$, hence, the equality holds.

If $L = \Sigma^\star$, then for any $w \in \Sigma^\star$ we have $w^{-1}L = \Sigma^\star = L$, hence:

$$\bigcup_{x\in\Sigma^\star} x^{-1}L \setminus \bigcap_{x\in\Sigma^\star} x^{-1}L = \Sigma^\star \setminus \Sigma^\star = \emptyset.$$

On the other hand, by definition (4), $\mathsf{D}(L) = \mathsf{suff}(L) \cap \mathsf{suff}(\bar{L}) = \Sigma^\star \cap \emptyset = \emptyset$, hence, again the equality holds.

Now let L be an arbitrary non-trivial language over Σ. Then, for any $w \in \mathsf{D}(L)$, by definition $\exists x, y \in \Sigma^\star$ such that $xw \in L$ and $yw \notin L$. Then $w \in x^{-1}L \ \wedge \ w \notin y^{-1}L$, therefore $\mathsf{D}(L) \subseteq \bigcup_{x\in\Sigma^\star} x^{-1}L \setminus \bigcap_{x\in\Sigma^\star} x^{-1}L$. Let $w \in \bigcup_{x\in\Sigma^\star} x^{-1}L \setminus \bigcap_{x\in\Sigma^\star} x^{-1}L$, then let x and y be such that $w \in x^{-1}L$ and $w \notin y^{-1}L$. Thus, $w \in \mathsf{D}(L)$. Hence, we conclude

$$\mathsf{D}(L) = \bigcup_{x\in\Sigma^\star} x^{-1}L \setminus \bigcap_{x\in\Sigma^\star} x^{-1}L.$$

□

Corollary 3.3. For a an arbitrary language L and $x, y \in \Sigma^\star$, $\mathsf{D}_L(x, y)$ is the symmetric difference of the correspondent quotients, $\mathsf{D}_L(x, y) = (x^{-1}L)\Delta(y^{-1}L)$.

To help the reader better understand how this operation looks like, we now present some more examples. In the first three examples we consider L a regular language and the correspondent minimal DFAs.

Example 3.4. If L is a regular language and $\mathsf{D}(L) = \{\varepsilon\}$, then we can only distinguish between final and non-final states, thus the minimal DFA of L has exactly two states corresponding to its two quotients.

Example 3.5. In this example we consider a family of languages L_n for which $\mathsf{D}(L_n) = \Sigma^\star$. Let $L_n = \mathcal{L}(A_n)$ for $3 \leq n$, with $A_n = \langle Q_n, \{0,1\}, \delta_n, 0, \{0\}\rangle$, where $Q_n = \{0, \ldots, n-1\}$, $\delta_n(i,0) = i+1 \mod n-1$, for $0 \leq i \leq n-1$ and $\delta_n(1,1) = 0$, $\delta_n(0,1) = 1$, $\delta_n(i,1) = i$, for $2 \leq i \leq n-1$. In Figure 1, we present A_5. Both symbols of the alphabet induce permutations on Q_n: 1 induces a transposition (2-cycle), and 0, an n cyclic permutation. It follows that, the transition semigroup of A_n is the symmetric group S_n of degree n, i.e., the set of all permutations of Q_n. We always have $\varepsilon \in \mathsf{D}(L)$, and every $w \in \Sigma^+$ induces a permutation on the states, for $0 \leq i, j \leq n-1$, $\delta(i,w) = i_w$ and $\delta(j,w) = j_w$, with $i_w \neq j_w$. Then, there must exist at least a pair (i,j), such that $w \in R_i$ and $w \notin R_j$, i.e., $w \in \mathsf{D}(L_n)$, thus $\mathsf{D}(L_n) = \Sigma^\star$. Bell et al. [1] studied those families of automata, and in particular, proved that they are uniformly minimal, i.e., minimal for every non-trivial choice of final states [20].

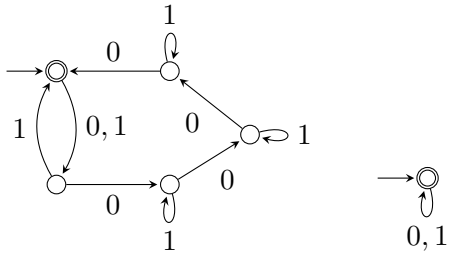

Figure 1. Automaton A_5 (left) and its distinguishability language, $\Sigma^\star$ (right).

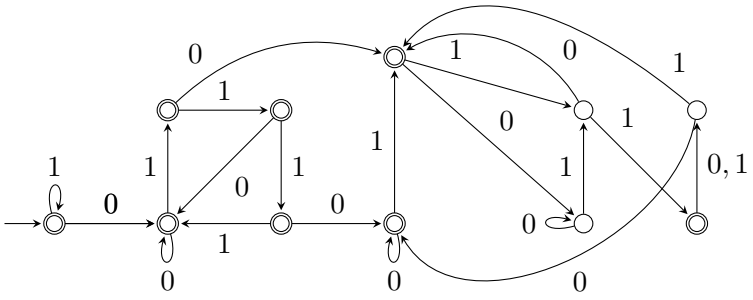

Figure 2. Example of an automaton with $\mathcal{L}(\mathcal{A}) \neq \mathsf{D}(\mathcal{L}(\mathcal{A}))$.

Example 3.6. Consider the automaton $\mathcal{A}$ in Figure 2. We have that $\mathcal{L}(\mathcal{A}) \neq \mathsf{D}(\mathcal{L}(\mathcal{A}))$, but $\mathsf{D}(\mathsf{D}(\mathcal{L}(\mathcal{A}))) = \mathsf{D}(\mathcal{L}(\mathcal{A}))$. The minimal automaton for $\mathsf{D}(\mathcal{L}(\mathcal{A}))$ is presented in Figure 3.

Lemma 3.7. There is a non-regular language L such that $\mathsf{D}(L)$ is regular.

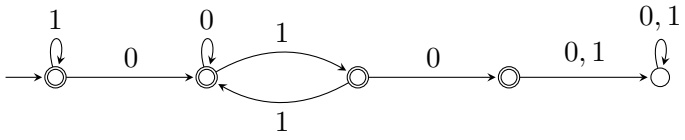

Figure 3. Example of an automaton $\mathcal{A}_1$ where $\mathsf{D}(\mathcal{L}(\mathcal{A}_1)) = \mathcal{L}(\mathcal{A}_1)$, i.e., the distinguishability language is the same as the language of the words it can distinguish.

Proof:
If we consider the language $L = \{a^{n^2} \mid n \geq 0\}$, we have that $\mathsf{D}(L) = \{a^n \mid n \geq 0\}$, which is regular. □

Example 3.8. Considering the language $L = \{a^n b^n c^n \mid n \geq 0\}$, we have that $\mathsf{D}(L) = \mathsf{suff}(L)$, because $\overline{L} \supseteq \{a^n b^n c^n w \mid w \neq \varepsilon\}$.

Example 3.9. A Dyck language is a language with k balanced parentheses and it is well-known to be context-free. If L is a Dyck language, then $\mathsf{D}(L) = \mathsf{suff}(L)$ and it is also context-free.

Example 3.10. Considering the context-free language $L = \Sigma^*\{abab\} \cup \{a^n b^n \mid n \geq 0\}$, we have that $\mathsf{D}(L) \neq \mathsf{suff}(L)$, because $abab \in \mathsf{suff}(L)$, but $abab \notin \mathsf{D}(L)$. Moreover, one can check that $\mathsf{D}(L)$ is also context-free (and non-regular).

Example 3.11. Considering the language $L = L(((0+1)(0+1))^\star(\varepsilon+1))$. In Figure 4 one can find, from left to right, a DFA that accepts L, one that accepts $\mathsf{D}(L) = (0 + 1^\star 10)^\star$, and one for $\mathsf{D}^n(L) = \varepsilon$, for $n \geq 2$.

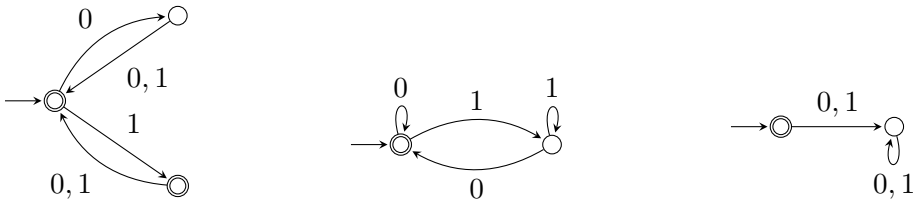

Figure 4. Automata for the languages L, $\mathsf{D}(L)$, and $\mathsf{D}^n(L)$, $n \geq 2$.

From the last example, we can see that the language $\mathsf{D}(L)$ contains the word 0110, but also the words 110, 10, and 0, which are all suffixes of 0110. This observation suggests that $\mathsf{D}(L)$ is suffix closed, which will be proved in the next theorem (for the general case again) later.

The following lemmas are useful in general.

Lemma 3.12. The suffix operator is an idempotent operator, i.e., for any set M, $\mathsf{suff}(\mathsf{suff}(M)) = \mathsf{suff}(M)$.

Proof:
For any M, by definition, $M \subseteq \mathsf{suff}(M)$. Hence, for any $w \in \mathsf{suff}(M)$, $w \in \mathsf{suff}(\mathsf{suff}(M))$, too, i.e., $\mathsf{suff}(\mathsf{suff}(M)) \supseteq \mathsf{suff}(M)$. On the other hand, for any $v \in \mathsf{suff}(\mathsf{suff}(M))$, by definition, there must exist x, y such that $yv \in \mathsf{suff}(M)$ and $xyv \in M$, which means $v \in \mathsf{suff}(M)$, too, i.e., $\mathsf{suff}(\mathsf{suff}(M)) \subseteq \mathsf{suff}(M)$. □

Lemma 3.13. If $L, M \subseteq \Sigma^\star$ are suffix-closed languages, then $\mathsf{suff}(L) \cap \mathsf{suff}(M) = \mathsf{suff}(L \cap M)$, and $\mathsf{suff}(L) \cup \mathsf{suff}(M) = \mathsf{suff}(L \cup M)$.

Proof:
It is obvious that the equality holds for union, and the inclusion $\mathsf{suff}(L \cap M) \subseteq \mathsf{suff}(L) \cap \mathsf{suff}(M)$, for intersection is true, too. If $w \in \mathsf{suff}(L) \cap \mathsf{suff}(M)$, then there exist $x, y \in \Sigma^\star$ such that $xw \in L$ and $yw \in M$. Because L and M are suffix closed, then $w \in L \cap M \subseteq \mathsf{suff}(L \cap M)$. □

Theorem 3.14. For any arbitrary language L, we have $\mathsf{suff}(\mathsf{D}(L)) = \mathsf{D}(L)$. Hence, the language $\mathsf{D}(L)$ is suffix closed.

Proof:
According to 4, we have $\mathsf{D}(L) = \mathsf{suff}(L) \cap \mathsf{suff}(\bar{L})$. Therefore, using Lemmas 3.12 and 3.13, we get $\mathsf{suff}(\mathsf{D}(L)) = \mathsf{suff}(\mathsf{suff}(L) \cap \mathsf{suff}(\bar{L})) = \mathsf{suff}(\mathsf{suff}(L)) \cap \mathsf{suff}(\mathsf{suff}(\bar{L})) = \mathsf{suff}(L) \cap \mathsf{suff}(\bar{L}) = \mathsf{D}(L)$. □

Due to the preceding theorem, $\mathsf{D}(L) = \mathsf{suff}(\mathsf{D}(L))$, hence, $\mathsf{suff}(\mathsf{D}(L)) \subseteq \mathsf{D}(L) \subseteq \mathsf{suff}(L)$ and $\mathsf{D}^2(L) \subseteq \mathsf{D}(L) \subseteq \mathsf{suff}(L)$. In general, for every $n \geq 1$, we have the following inclusion

$$\mathsf{D}^{n+1}(L) \subseteq \mathsf{D}^n(L). \tag{5}$$

Consequently, we may ask if this hierarchy is infinite or not, in other words, we may ask if for any language L, there exists an $n \geq 0$ such that $\mathsf{D}^{n+1}(L) = \mathsf{D}^n(L)$.

Example 3.15. Consider the language $L = \mathcal{L}(\mathcal{A})$, where $\mathcal{A}$ is given in Figure 5, on the left. For the language L, we have that $L \neq \mathsf{D}(L)$ and $\mathsf{D}(L) \neq \mathsf{D}^2(L) = \mathsf{D}^n(L)$, for $n \geq 2$. The minimal automaton for $\mathsf{D}^2(L)$ is depicted on the right. The minimal automaton for $\mathsf{D}(L)$ has 7 states.

In the following result, we prove that the iteration of D operations always reaches a fixed point.

Theorem 3.16. Let $L \subseteq \Sigma^\star$ be an arbitrary language. Then we have that $\mathsf{D}^3(L) = \mathsf{D}^2(L)$.

Proof:
We have the following equalities:

$$\mathsf{D}^2(L) \quad = \quad \mathsf{D}(\mathsf{D}(L)) = \mathsf{suff}(\mathsf{D}(L)) \cap \mathsf{suff}(\overline{\mathsf{D}(L)}) = \mathsf{D}(L) \cap \mathsf{suff}(\overline{\mathsf{D}(L)}). \tag{6}$$

Now, computing the next iteration of D, we get $\mathsf{D}^3(L) = \mathsf{suff}(\mathsf{D}^2(L)) \cap \mathsf{suff}(\overline{\mathsf{D}^2(L)})$.
Using (6) and Lemma 3.13, we obtain the equalities

$$\mathsf{suff}(\overline{\mathsf{D}^2(L)}) = \mathsf{suff}(\overline{\mathsf{D}(L) \cap \mathsf{suff}(\overline{\mathsf{D}(L)})}) \quad = \quad \mathsf{suff}(\overline{\mathsf{D}(L)}) \cup \mathsf{suff}(\overline{\mathsf{suff}(\overline{\mathsf{D}(L)})}).$$

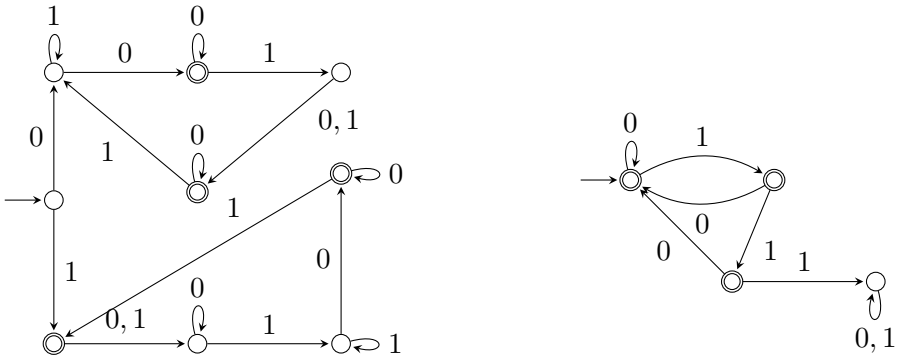

Figure 5. Example of a language L with $\mathsf{D}(L) \neq \mathsf{D}^2(L) = \mathsf{D}^n(L)$, for $n \geq 3$. On the left, a DFA for L and on the right a DFA for $\mathsf{D}^2(L)$.

Because $\mathsf{D}^2(L)$ is a suffix-closed language, it follows that

$$\begin{aligned}
\mathsf{D}^3(L) &= \mathsf{D}^2(L) \cap (\mathsf{suff}(\overline{\mathsf{D}(L)}) \cup \mathsf{suff}(\overline{\mathsf{suff}(\overline{\mathsf{D}(L)})})) \\
&= (\mathsf{D}^2(L) \cap \mathsf{suff}(\overline{\mathsf{D}(L)})) \cup (\mathsf{D}^2(L) \cap \mathsf{suff}(\overline{\mathsf{suff}(\overline{\mathsf{D}(L)})}))) \\
&= \mathsf{D}^2(L) \cup (\mathsf{D}^2(L) \cap \mathsf{suff}(\overline{\mathsf{suff}(\overline{\mathsf{D}(L)})}))) = \mathsf{D}^2(L).
\end{aligned}$$

□

The following results give characterizations for some languages that are fixed points for D.

Lemma 3.17. Given any language L, if L has $\emptyset$ as a quotient, then $\mathsf{D}(L) = \mathsf{suff}(L)$.

Proof:
Because $z^{-1}L = \emptyset$, for some word z, we have $\mathsf{suff}(\overline{L}) = \Sigma^\star$. □

This lemma makes the following result immediate.

Theorem 3.18. L is a fixed point for D, i.e., $\mathsf{D}(L) = L$, if and only if L is suffix closed and has $\emptyset$ as one of its quotients.

Corollary 3.19. Let L be any language, then $\mathsf{D}(L)$ has $\emptyset$ as a quotient if and only if $\mathsf{D}^2(L) = \mathsf{D}(L)$.

Note that the condition L to be suffix closed is not sufficient to ensure that L has $\emptyset$ as a quotient. For that, it is enough to consider the language given by $0^\star + 0^\star 1(1 + 00^\star 1)^\star$. However, if L is a D fixed point, the implication holds at least for regular languages.

Theorem 3.20. Let L be a regular language. If $\mathsf{D}(L) = L$, then L has $\emptyset$ as a quotient.

Proof:
Let L be a regular language that is fixed point for D, thus L is suffix closed and

$$(\forall w \in L)(\exists u \in \Sigma^\star)(uw \notin L). \tag{7}$$

Assume that L does not have $\emptyset$ as quotient, i.e.,

$$(\forall w \in \Sigma^\star)(\exists v \in \Sigma^\star)(wv \in L). \tag{8}$$

Let $w \notin L$ ($\Sigma^\star$ is not a fixed point for D). Thus, by (8) there exists a $v_0 \in \Sigma^\star$ such that $wv_0 \in L$ and because L is suffix closed, it follows that $v_0 \in L$. Using (7), there exists u_0 such that $u_0wv_0 \notin L$. Using the same reasoning, we can find $u_1 \in \Sigma^\star$ and $v_1 \in L$ such that

$$u_0wv_0v_1 \in L \text{ and } u_1u_0wv_0v_1 \notin L.$$

The word wv_0v_1 distinguishes u_0 from u_1u_0, thus these words cannot belong to the same quotient. Suppose that we have iterated n times this process having

$$u_{n-1}\cdots u_0wv_0\cdots v_n \in L \text{ and } u_n\cdots u_0wv_0\cdots v_n \notin L,$$

with all $u_i\cdots u_0$ belonging to distinct quotients. We can apply this process one more time, obtaining

$$u_n\cdots u_0wv_0\cdots v_{n+1} \in L \text{ and } u_{n+1}\cdots u_0wv_0\cdots v_{n+1} \notin L.$$

It is easy to see that the word $wv_0\cdots v_{n+1}$ distinguishes $u_{n+1}\cdots u_0$ from any of the previous words $u_i\cdots u_0$ (with $i \leq n$) because $u_i\cdots u_0wv_0\cdots v_{n+1}$ is a suffix of $u_n\cdots u_0wv_0\cdots v_{n+1} \in L$. Thus, the number of L quotients cannot be finite, a contradiction. □

By contraposition over the last result, we get that a language L with all its quotients being non-empty cannot be a fixed point for D. We know that if a language L is such that $\Sigma^\star = \mathsf{suff}(\overline{L})$, then $\emptyset$ must be one of the quotients of L. In Examples 3.5–3.11 and Example 3.15, we have languages L with all quotients being non-empty and $L \neq \mathsf{D}(L)$. Considering Theorem 3.20 and Theorem 3.16, given any regular language L we can iterate D at most two times to obtain a language that has $\emptyset$ as a quotient.

For a finite language L, it follows from Lemma 3.17 that the distinguishability language of L coincides with the set of all suffixes of L, therefore, $\mathsf{D}(L)$ is a fixed point of the D operator.

Corollary 3.21. If L is a finite language, then $\mathsf{D}(L) = \mathsf{suff}(L)$.

The minimal DFA that represents the set of suffixes of a finite language L is called the *suffix automaton*, and several optimized algorithms for its construction were studied in the literature. Thus, we can use an algorithm for building the suffix automaton in order to obtain $\mathsf{D}(L)$. Recently, Mohri et al. [19] gave new upper bounds on the number of states of the suffix automaton as a function of the size of the minimal DFA of L, as well as other measures of L. In Section 4, we study the state complexity of $\mathsf{D}(L)$ as a function of the state complexity of L, for any general regular language L. The state complexity of D for finite languages was studied by Câmpeanu et al. [10]. In Section 7, we generalize the results on the characterization of the distinguishability operation, defining languages that distinguish between pairs of different right and two-sided quotients.

4. State Complexity

Let L be a regular language. By definition, $\mathsf{D}(L)$ can be obtained using the suffix operator, complement and intersection, therefore, it is a result of combining three operations, two unary and one binary. We would like to estimate the state complexity of the D operation and check if the upper bound is tight. We recall that the state complexity of an operation is the worst-case state complexity of a language resulting from that operation, as a function of the state complexities of the operands.

The following theorem shows the construction for $\mathsf{D}(L)$, in case L is recognized by a DFA.

Theorem 4.1. Let $\mathcal{A} = (Q, \Sigma, \delta, i, F)$ be a reduced DFA recognizing a (nontrivial) language L. Then $\mathcal{A}_d = (Q_d, \Sigma, \delta_d, Q, F_d)$ is a DFA that accepts $\mathsf{D}(L)$, where

- $Q_d = \{S \mid S \subseteq Q\}$,
- for $a \in \Sigma$ and $S \in Q_d$, $\delta_d(S, a) = \{\delta(q, a) \mid q \in S\}$,
- $F_d = \{S \mid S \cap F \neq \emptyset \text{ and } S \cap (Q \setminus F) \neq \emptyset\}$.

Proof:
Considering that $\mathsf{D}(L) = \mathsf{suff}(L) \cap \mathsf{suff}(\overline{L})$, we can use the usual subset construction for $\mathsf{suff}(L)$ to build an NFA with the same transition function as $\mathcal{A}$, and all its states being initial. For $\mathsf{suff}(\overline{L})$, the corresponding NFA will be the same, but flipping the finality for all the states. Because both operands share the same structure, the DFA corresponding to the intersection will be the DFA resulting from the subset construction considering a suitable set of final states (they must contain at least one final state and a non-final one). □

Let $\mathcal{A} = (Q, \Sigma, \delta, i, F)$ be the minimal DFA recognizing L with $|Q| = n$. Let $Q = \{0, \ldots, n-1\}$ and R_i, $0 \leq i \leq n-1$, be the left quotients of L (possibly including the empty set). From Theorem 3.2, we have:

$$\mathsf{D}(L) = \bigcup_{i \in Q} R_i \setminus \bigcap_{i \in Q} R_i = \left(\bigcup_{i \in Q} R_i\right) \cap \overline{\left(\bigcap_{i \in Q} R_i\right)} = \overline{\left(\bigcap_{i \in Q} \overline{R_i}\right) \cup \left(\bigcap_{i \in Q} R_i\right)}. \tag{9}$$

In the following we identify the states of $\mathcal{A}$ with the corresponding left quotients. Instead of using traditional techniques to prove the correctness of tight upper bounds of operational state complexity, here we consider a method based on the *atoms* of regular expressions. Using this approach, we aim to provide yet another piece of evidence for their broad applicability.

Brzozowski and Tamm introduced the notion of atoms of regular languages in [6] and studied their state complexity in [7]. An *atom* of a regular language L with n quotients $R_0, \ldots, R_{n-1}$ is a non-empty intersection $K_0 \cap \cdots \cap K_{n-1}$, where each K_i is a quotient R_i, or its complement $\overline{R_i}$. Atoms of L are partitions of $\Sigma^\star$. In particular, $A_Q = \bigcap_{i \in Q} R_i$ ($A_\emptyset = \bigcap_{i \in Q} \overline{R_i}$) is an atom with zero complemented (uncomplemented) quotients. In [7] it was proved that the state complexity of both those atoms is $2^n - 1$. Using similar arguments, we prove the following theorem.

Theorem 4.2. If a regular language L has a minimal DFA with $n \geq 2$ states, then $sc(\mathsf{D}(L)) \leq 2^n - n$.

Proof:
Let $\mathcal{A} = (Q, \Sigma, \delta, i, F)$ be the minimal DFA recognizing L with $|Q| = n$. Then $Q = \{0, \ldots, n-1\}$, and let R_i, $0 \leq i \leq n-1$ be the (left) quotients of L. Using Equation (9), every quotient $w^{-1}\mathsf{D}(L)$ of $\mathsf{D}(L)$, for $w \in \Sigma^\star$, is given by:

$$w^{-1}\mathsf{D}(L) = \left(\bigcup_{i \in Q} w^{-1}R_i\right) \cap \overline{\left(\bigcap_{i \in Q} w^{-1}R_i\right)},$$

where all $w^{-1}R_i$, $0 \leq i \leq n-1$, are also quotients of L, and they may not be distinct. Considering all non-empty subsets of quotients of L, there would be at most 2^n quotients of $\mathsf{D}(L)$. However, all subsets, R_j, with exactly one element will lead to the empty quotient[1]. Thus, $sc(\mathsf{D}(L)) \leq 2^n - n$. □

Brzozowski [3] presented a family of languages U_n which provides witnesses for the state complexity of several individual and combined operations over regular languages. Brzozowski and Tamm [7] proved that U_n was also a witness for the worst-case state complexity of atoms. This family is defined as follows. For each $n \geq 2$, we construct the DFAs $D_n = (\{0, \ldots, n-1\}, \{0, 1, 2\}, \delta, 0, \{n-1\})$, where $\delta(i, 0) = i + 1 \mod n$, $\delta(0, 1) = 1$, $\delta(1, 1) = 0$, $\delta(i, 1) = i$ for $i > 1$, $\delta(i, 2) = i$ for $0 \leq i \leq n-2$, and $\delta(n-1, 2) = 0$. We denote by U_n the language accepted by D_n, i.e.,

$$U_n = \mathcal{L}(D_n). \tag{10}$$

We show that U_n is also a witness for the lower bound of the state complexity of $\mathsf{D}(L)$. First, observe that the automata D_n, $n \geq 2$ are minimal. In Figure 6, we present D_4.

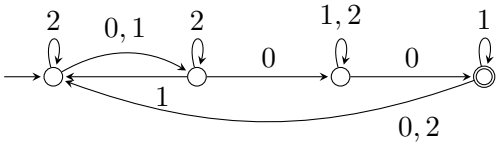

Figure 6. Universal witness D_4.

Next, we give the lower bound for the number of states of a DFA accepting $\mathsf{D}(U_n)$.

Theorem 4.3. For $n \geq 2$, the minimal DFA accepting $\mathsf{D}(U_n)$ has $2^n - n$ states.

Proof:

Let $A_n = (R_0 \cap \ldots \cap R_{n-1})$, and $A_\emptyset = (\overline{R_0} \cap \ldots \cap \overline{R_{n-1}})$, be the two atoms of U_n as above, where R_i are its quotients $0 \leq i \leq n-1$. Then $\mathsf{D}(U_n) = \overline{A_n \cup A_\emptyset}$. Brzozowski and Tamm proved that $sc(A_n) = sc(A_\emptyset) = 2^n - 1$. Applying the construction given in Theorem 4.1 to $\mathsf{D}(U_n)$, and noting that a regular language and its complement have the same state complexity, we obtain the upper bound. □

[1]Because all transitions are deterministic from singleton sets of states only singleton sets will be reached, which will be either a final state or a non-final one.

If $sc(L) = 1$, as noted in the proof of Theorem 3.2, we have that $sc(\mathsf{D}(L)) = 1$. If L has $\emptyset$ as a quotient, by Lemma 3.17, the upper bound for $sc(\mathsf{D}(L))$ coincides with the one for $\mathsf{suff}(L)$, i.e., it is 2^{n-1} if $sc(L) = n$, cf. [5]. This upper bound is achieved by the family of languages represented in Figure 7.

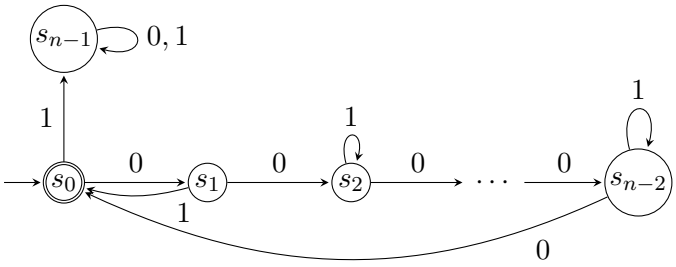

Figure 7. Witness family for $sc(\mathsf{D}(L))$ when L has $\emptyset$ as a quotient.

Having considered some properties of the distinguishability language, we would like to select only the set of minimal words that distinguishes between distinct quotients. Obviously, this is a subset of $\mathsf{D}(L)$, and in the following section we study some of its properties.

5. Minimal Distinguishable Words

An even more succinct language distinguishing all different quotients of a regular language, in fact a finite one, can be obtained if we consider only the shortest word that distinguishes each pair of quotients.

Definition 5.1. Let L be an arbitrary language, and assume we have an order over the alphabet Σ. If $x, y \in \Sigma^\star$ and $x \not\equiv_L y$, we define

$$\underline{\mathsf{D}}_L(x, y) = \min \{w \mid w \in \mathsf{D}_L(x, y)\},$$

where minimum is considered with respect to the quasi-lexicographical order. In case $x \equiv_L y$, $\underline{\mathsf{D}}_L(x, y)$ is undefined. We can observe that if $x \not\equiv_L y$, $\underline{\mathsf{D}}_L(x, y) = \min(x^{-1}L \Delta y^{-1}L)$.

The set of minimal words distinguishing quotients of a language L is

$$\underline{\mathsf{D}}(L) = \{\underline{\mathsf{D}}_L(x, y) \mid x, y \in \Sigma^*, x \not\equiv_L y\}.$$

Example 5.2. We present a few simple cases. Similar to the D operator, we have the equalities: $\underline{\mathsf{D}}(\Sigma^\star) = \underline{\mathsf{D}}(\emptyset) = \emptyset$ and $\underline{\mathsf{D}}(\{\varepsilon\}) = \{\varepsilon\}$. In case $0 \in \Sigma$, $\underline{\mathsf{D}}(\{0\}) = \mathsf{D}(\{0\}) = \{0, \varepsilon\}$, and $\underline{\mathsf{D}}(\{0^n\}) = \mathsf{D}(\{0^n\}) = \{0^i \mid 0 \leq i \leq n\}$, for $n \geq 2$.

Example 5.3. Consider the language L in the proof of Lemma 3.7. We have that $\underline{\mathsf{D}}(L) = \{0^n \mid n \geq 0\}$, because 0^n is the minimal word distinguishing between 0^{k^2-n} and 0^{k^2-n-1}, where k is such that there is no perfect square in the set $\{k^2 - n - 1, k^2 - n, \ldots, k^2 - 1\}$, for $n \geq 0$.

Example 5.4. Consider the language L of Example 3.15. We have the following equalities $\underline{\mathsf{D}}(L) = \{\varepsilon, 0, 1, 01, 11\}$, $\underline{\mathsf{D}}(\mathsf{D}(L)) = \{\varepsilon, 1, 01, 11\}$, and $\underline{\mathsf{D}}(\mathsf{D}^2(L)) = \{\varepsilon, 1, 11\}$.

The previous example suggests that also $\underline{\mathsf{D}}(L)$ is suffix closed.

Theorem 5.5. If L is an arbitrary language, then $\underline{\mathsf{D}}(L)$ is suffix closed.

Proof:
Let $w \in \underline{\mathsf{D}}(L)$, and let $w = uv$, with $u, v \in \Sigma^\star$. Because $w \in \underline{\mathsf{D}}(L)$, we can find two other words, $x, y \in \Sigma^\star$, such that $xw \in L$ and $yw \notin L$, i.e., $xuv \in L$ and $yuv \notin L$. It follows that $v \in \mathsf{D}_L(xu, yu)$. Since $v \in \mathsf{D}_L(xu, yu)$, there exists $v' \in \underline{\mathsf{D}}_L(xu, yu)$ and $v' \preceq v$. Hence, $uv' \preceq uv$ and $uv' \in \mathsf{D}_L(x, y)$, which implies that $w = uv \preceq uv'$. Then we must have $uv' = uv$, which implies that $v = v'$, hence, $\underline{\mathsf{D}}_L(xu, yu) \in \underline{\mathsf{D}}(L)$. □

The next result gives an upper bound for the number of elements of $\underline{\mathsf{D}}(L)$.

Theorem 5.6. If L is a regular language with state complexity $n \geq 2$, then $|\underline{\mathsf{D}}(L)| \leq n - 1$.

Proof:
For any three sets A, B and C we have the equality $(A \Delta B) \Delta (B \Delta C) = A \Delta C$. Therefore, we can distinguish any pair from n distinct sets with at most $n - 1$ elements. To prove the theorem it is enough to choose the minimal words satisfying the above conditions, since the n quotients of L are all distinct (their symmetric difference is non-empty). □

Now, we prove that the upper bound is reached.

Theorem 5.7. The bound $n - 1$ for the size of $\underline{\mathsf{D}}(L)$, for a regular language L with state complexity $n \geq 2$, is tight.

Proof:
Consider again the family of languages U_n, described by Equation (10). For each state $0 \leq i \leq n - 1$ of D_n, let R_i be the corresponding quotient. It is easy to see that the minimal word for each quotient R_i is 0^{n-i-1}, and we can disregard the largest one of those. □

The previous proof, we have that $sc(\underline{\mathsf{D}}(U_n)) = sc(\{0^i \mid 0 \leq i \leq n-2\}) = n$. This shows that for regular languages with state complexity n, a lower bound for the state complexity of $\underline{\mathsf{D}}$ is n.

Now we consider the iteration of the $\underline{\mathsf{D}}$ operator. Because $\underline{\mathsf{D}}(L) \subseteq \mathsf{D}(L)$, $\underline{\mathsf{D}}(L) \subseteq \mathsf{suff}(L)$, and $\underline{\mathsf{D}}(L)$ is suffix closed, it follows that $\underline{\mathsf{D}}^2(L) \subseteq \underline{\mathsf{D}}(L)$, and, in general,

$$\underline{\mathsf{D}}^{n+1}(L) \subseteq \underline{\mathsf{D}}^n(L), \text{ for all } n \geq 1. \tag{11}$$

By the finiteness of $\underline{\mathsf{D}}(L)$, it follows that there exists $n \geq 0$ such that $\underline{\mathsf{D}}^{n+1}(L) = \underline{\mathsf{D}}^n(L)$. For instance, considering the family of languages U_n defined by Equation (10), we have that $\underline{\mathsf{D}}^2(U_n) = \underline{\mathsf{D}}(U_n)$.

Contrary to the hierarchy for $\mathsf{D}(L)$, where the fixed point is reached for $n = 2$, in the case of $\underline{\mathsf{D}}(L)$ we have that for any $n \geq 0$, there is a language for which the fixed point is reached after n iterations of $\underline{\mathsf{D}}$.

Theorem 5.8. Given a regular language L with state complexity n, the fixed point of $\underline{\mathsf{D}}^i(L)$, $0 \leq i \leq n-2$ is reached for some $i \leq n-2$.

Proof:
Because $\underline{\mathsf{D}}(L)$ is suffix closed, $\varepsilon \in \underline{\mathsf{D}}^i(L)$ for all every $i \geq 1$, thus any automaton recognizing $\underline{\mathsf{D}}^i(L)$ has at least 2 states. By Theorem 5.6, $sc(\underline{\mathsf{D}}(L)) \leq n-1$; thus, taking into account that $\underline{\mathsf{D}}^i(L)$ is suffix closed for $i \geq 1$, we either must get the same set or a smaller set, in that case losing at least one element at each iteration. Hence, $i \leq n-2$. □

If in the previous theorem we have established an upper bound for the number of iterations of the $\underline{\mathsf{D}}$ operator necessary to reach a fixed point, in the next one we show that the upper bound can be reached.

Theorem 5.9. For all $n \geq 3$, there exists a regular language L_n, with $sc(L_n) = n$, such that

i) $\underline{\mathsf{D}}^{m-1}(L_n) \neq \underline{\mathsf{D}}^m(L_n)$, for all $m < n-2$, and

ii) $\underline{\mathsf{D}}^{n-2}(L_n) = \underline{\mathsf{D}}^{n-1}(L_n)$.

Proof:
Consider the family of languages $W_m = \mathsf{suff}(0^m1) = \left\{0^i1 \mid 0 \leq i \leq m\right\} \cup \{\varepsilon\}$, $m \geq 0$. Then $sc(W_m) = m+3$ and $\underline{\mathsf{D}}(W_m) = \left\{0^i1 \mid 0 \leq i \leq m-1\right\} \cup \{\varepsilon\} = W_{m-1}$. Because $W_0 = \{1, \varepsilon\}$ is a fixed point for $\underline{\mathsf{D}}$, i.e., $\underline{\mathsf{D}}(W_0) = W_0$, we obtain

1. $\underline{\mathsf{D}}^{m-1}(W_{n-3}) \neq \underline{\mathsf{D}}^m(W_{n-3})$, for all $m < n-2$, and
2. $\underline{\mathsf{D}}^{n-2}(W_{n-3}) = \underline{\mathsf{D}}^{n-1}(W_{n-3})$.

Hence, we can just take $L_n = W_{n-3}$. □

In the next section we use $\underline{\mathsf{D}}(L)$ to recover L as an l-cover language for $L \cap \Sigma^{\leq l}$.

6. Using Minimal Distinguishability Words to Recover the Original Language

In Section 1 we claim that for any regular language L, there exists a constant l, such that having the distinguishability language $\underline{\mathsf{D}}(L)$, we can recover the original language L, if we know all the words in L of length less than or equal to l, i.e., the set $L \cap \Sigma^{\leq l}$.

Thus, a positive learning procedure can be designed to recover the original language, L, from every pair $(\underline{\mathsf{D}}(L), l)$ such that l is large enough. It is obvious that if L is a regular language, and $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ a finite automaton recognizing L, i.e., $L = \mathcal{L}(\mathcal{A})$, then L is always an l-cover language for $L \cap \Sigma^{\leq l}$. Thus, the goal is to determine the language L as a unique l-cover language for $L \cap \Sigma^{\leq l}$.

Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$, and L be a regular language such that $L = \mathcal{L}(\mathcal{A})$. The automaton $\mathcal{A}$ is minimal if for any two states $p, q \in Q$, $p \neq q$, we can find a word to distinguish between them. Hence, $\mathcal{A}$ is minimal if we can find a word $w \in \underline{\mathsf{D}}(L)$ such that it distinguishes between p and q, thus the words $x_{\mathcal{A}}(p)$ and $x_{\mathcal{A}}(q)$ are distinguishable by some word in $w \in \underline{\mathsf{D}}(L)$.

Let us consider the Myhill-Nerode equivalence induced by L, $\equiv_L$. Two words x_1 and x_2 are equivalent, with respect to $\equiv_L$, if and only if there exists $w \in \underline{\mathsf{D}}(L)$ such that $x_1 w \in L$ iff $x_2 w \in L$. Thus, by generating all words in the language of length $\max(|x_1|, |x_2|) + \max\{|w| \mid w \in \underline{\mathsf{D}}(L)\}$, we can decide after a finite number of steps if $x_1 \equiv_L x_2$.

Therefore, the following algorithm can select all words $x_{\mathcal{A}}(p)$ for a minimal DFA recognizing L:

1: $n \leftarrow 1, x \leftarrow \varepsilon, x_{\mathcal{A}}(n) \leftarrow x, Q \leftarrow \{n\}, l \leftarrow 0$
2: $y \leftarrow \mathsf{Succ}(x)$, where $\mathsf{Succ}(x)$ is the next word for the quasi-lexicographical order
3: **while** $|y| \leq l + 1$ **do**
4: **if** $y \not\equiv x_{\mathcal{A}}(q), \forall q \in Q$ **then**
5: $n \leftarrow n + 1, x_{\mathcal{A}}(n) \leftarrow y, Q \leftarrow Q \cup \{n\}, l \leftarrow |y|$
6: **else**
7: $\delta(p, a) \leftarrow q$, where $y = za \equiv_L x_{\mathcal{A}}(q), z = x_{\mathcal{A}}(p), a \in \Sigma$
8: **end if**
9: $x \leftarrow y$, $y \leftarrow \mathsf{Succ}(x)$
10: **while** y is unreachable and $|y| \leq l + 1$ **do** ▷ i.e., $y = x_{\mathcal{A}}(q)wa$, $x_{\mathcal{A}}(q)w \equiv_L x_{\mathcal{A}}(p)$, for some $p, q \in Q$, $w \in \Sigma^+$, $a \in \Sigma$
11: $x \leftarrow y$, $y \leftarrow \mathsf{Succ}(x)$
12: **end while**
13: **end while**
14: Set as final all $q \in Q$ such that $x_{\mathcal{A}}(q) \in L$

We observe that if all the words of length $|x| = l + 1$ are equivalent with some shorter words, then all the words of length greater than $l + 1$ will be declared unreachable by the previous algorithm, and no new transition can be added. On the other hand, $\delta(q, a)$ is always defined, as $x_{\mathcal{A}}(q)a = x_{\mathcal{A}}(p)$, for some $p \not\equiv q$, or $x_{\mathcal{A}}(q)a \equiv x_{\mathcal{A}}(p)$, and $\delta(q, a) = p$. All the states in the above construction correspond to distinguishable words $x_{\mathcal{A}}(q)$, thus the DFA is minimal.

Note that when testing in quasi-lexicographical order if a new word in $\Sigma^\star$ is equivalent with previously distinct ones, will prune the branches corresponding to equivalent words, and testing done in line 10 needs only to test if y is on a cut-out branch. Because all words of length greater than $n = sc(L)$ must be equivalent with some word of length less than n, it is enough to test only words of length at most $sc(L)+1$. In order to conduct the equivalence test, it is enough to generate all words of length $n+d+1$, where $d = \max\{|w| \mid w \in \underline{\mathsf{D}}(L)\}$. Hence, all steps in the algorithm are well defined and they are only executed a finite number of times, thus it will produce a minimal DFA after a finite number of steps.

Therefore, we just have proved the following theorem:

Theorem 6.1. Let L be a regular language and $L_D = \underline{\mathsf{D}}(L)$. If we have an algorithm to generate all words in the language L up to a given length m, then we have an algorithm to compute

i) A number l, such that L is an l-cover language for $L \cap \Sigma^{\leq l}$;

ii) A minimal DFA A for L, which is at the same time, an l-DFCA for $L \cap \Sigma^{\leq l}$.

A crucial role for producing the algorithm is the fact that testing the equivalence of two words can be done using a finite number of steps, and this is possible if we know $\underline{\mathsf{D}}(L)$. However, it is not known

if an equivalent procedure can be obtained if we know only $\mathsf{D}(L)$, as it is possible to have two languages L_1,L_2 such that they share the same D languages, but different $\underline{\mathsf{D}}$ languages.

For example, we can take $L_1 = \{w \mid w = aaxba^n, x \in \{a,b\}^*, n \geq 0\}$ and $L_2 = \{w \mid w = bbxab^n, x \in \{a,b\}^*, n \geq 0\}$. We have that $\underline{\mathsf{D}}(L_1) = \{b, ab, e, aab\}$, $\underline{\mathsf{D}}(L_2) = \{a, e, ba, bba\}$, and $\mathsf{D}(L_1) = \mathsf{D}(L_2) = \Sigma^\star$. This example suggests why knowing $\mathsf{D}(L)$ may not be enough.

In the next section we check under what conditions the results obtained so far can be generalized.

7. Boolean Operations and Closure

In Section 3 we used Boolean operations and the suffix operation to compute the distinguishability language. The suffix operation has the following properties:

i) $\mathsf{suff}(\emptyset) = \emptyset$;

ii) $L \subseteq \mathsf{suff}(L)$;

iii) $\mathsf{suff}(\mathsf{suff}(L)) = \mathsf{suff}(L)$;

iv) $\mathsf{suff}(L_1 \cup L_2) = \mathsf{suff}(L_1) \cup \mathsf{suff}(L_2)$,

thus, it is a closure operator.

If we consider the distinguishability operation as a unary operation on r languages, we can see that it is obtained by applying finitely many times a closure operator and Boolean operations. The Closure-Complement Kuratowski Theorem [18] says that using one set, one can obtain at most 14 distinct sets using finitely many times one closure operator and the complement operation. For the case of regular languages, Brzozowski et al. [4] determine the number of languages that can be obtained by applying finitely many times the Kleene closure and complement. However, the corresponding property (iv) is not satisfied by the Kleene closure.

In this section we analyze the case of closure operators and Boolean operations, and ask if applying them finitely many times, we can still obtain only finitely many sets, or what is a necessary condition to obtain only finitely many sets.

In order to do this, we need to prove some technical lemmata. In the following, M denotes a nonempty set.

Lemma 7.1. Let $N = \{\mathcal{A}_1, \ldots, A_n\}$, where $\mathcal{A}_i \in 2^M$, $1 \leq i \leq n$. Then the free algebra $(N, \cup, \cap, \bar{\cdot}, \emptyset, N)$ has a finite number of elements.

Proof:

All expressions can be reduced to the disjunctive normal form, and we only have finitely many such formulae. □

Lemma 7.2. Let $c_1, c_2 : 2^M \longrightarrow 2^M$ be two closure operators such that c_1 and c_2 commute, i.e., $c_1 \circ c_2 = c_2 \circ c_1$. Then the composition $c = c_2 \circ c_1$ is also a closure operator.

Proof:
Let us verify the properties of a closure operator, thus if $L, L_1, L_2 \in 2^M$, we have:

i) $c(\emptyset) = (c_2 \circ c_1)(\emptyset) = c_2(c_1(\emptyset)) = \emptyset$;

ii) $L \subseteq c_2(L) \subseteq c_1(c_2(L)) = (c_2 \circ c_1)(L) = c(L)$;

iii) $c(c(L)) = (c_2 \circ c_1)((c_2 \circ c_1)(L)) = (c_2 \circ c_1)((c_1 \circ c_2)(L)) = (c_2 \circ c_1 \circ c_1 \circ c_2)(L) = (c_2 \circ c_1 \circ c_2)(L) = (c_2 \circ c_2 \circ c_1)(L) = (c_2 \circ c_1)(L) = c(L)$;

iv) $c(L_1 \cup L_2) = (c_2 \circ c_1)(L_1 \cup L_2) = c_2(c_1(L_1 \cup L_2)) = c_2(c_1(L_1) \cup c_1(L_2)) = c_2(c_1(L_1)) \cup c_2(c_1(L_2)) = c(L_1) \cup c(L_2)$.

□

In general, not all closure operators commute, for example, $N_0, N_1 : 2^{\mathbb{N}} \longrightarrow 2^{\mathbb{N}}$ defined by $N_0(A) = A \cup \{x \in \mathbb{N} \mid \exists k > 0, x = 2k \text{ if } 2k-1 \in A\}$, $N_1(A) = A \cup \{x \in \mathbb{N} \mid \exists k >= 0, x = 2k+1 \text{ if } 2k \in A\}$, do not commute one with each other, as N_0 adds all the even numbers that are successors of elements in the set A, and N_1 adds all the odd numbers that are successors of elements in the set A. Applying the closure operators alternatively to a finite set A, we always obtain a new set.

The next result is well known for the behaviour of closure operators when applied to an intersection of two other sets.

Lemma 7.3. Let c be a closure operator on 2^M. If $L_1, L_2 \in 2^M$ are closed subsets, then $c(L_1) \cap c(L_2) = c(L_1 \cap L_2)$.

Assume we have a finite number of sets $L_1, \ldots, L_m$. Using closure and complement for each set L_i, $1 \leq i \leq m$, we can obtain a finite number of sets [18], say $M_1, \ldots, M_l$. Now consider a Boolean expression using $M_1, \ldots, M_l$. Because we can transform all these Boolean expressions in disjunctive normal form, the number of Boolean expressions over $M_1, \ldots, M_l$ is finite. Applying the closure operator to such an expression will commute with union, and the other sets are in the form $c(M_{i_1} \cap \cdots \cap M_{i_k})$. If all M_{i_j} $1 \leq j \leq k$ are closed sets, then $c(M_{i_1} \cap \cdots \cap M_{i_k})$ is a conjunction of some other sets $M_{j_1}, \ldots, M_{j_k}$, $1 \leq j_i \leq l$. Otherwise, if a set M_{i_j} $1 \leq j \leq k$ is not closed, we may obtain new sets, as we can see from the following example: if $L_1 = \{aaaa, abaabbaa, b\}$, $L_2 = \{bbb, baaab, aa\}$, where $c = \mathsf{suff}$, then $c(L_1 \cap c(L_2)) \cap L_2 \neq c(L_1) \cap L_2$.

This suggests that if a unary operation that combines Boolean operations and closure operators is repeatedly applied to a set and we first apply the closure operator to the set and its complement, then we use other Boolean operations or the closure operator finitely many times, we will always obtain finitely many sets. Thus, we have just proved the following lemma:

Lemma 7.4. Let c be a closure operator on 2^M. If $\mathsf{O} : 2^M \longrightarrow 2^M$ is defined as a union and intersections over $c(L)$ and $c(\overline{L})$, for $L \in 2^M$, then

1. for every set A, $\mathsf{O}(A) = c(B)$, for some $B \in 2^M$;
2. any iteration of O will produce a finite number of sets;
3. if $\mathsf{O}(L) \subset L$, for all L, then O has a fixed point.

Of course, if we have more than one closure operator, and we want to obtain finitely many sets, we must first apply one closure operator to the collection of sets and their complements, then all the other Boolean operators and closure operator again. In this way, we have guaranteed that we can only obtain finitely many sets. In case the operation defined this way is monotone and bounded, it will have a fixed point.

In particular, we have a generalization of Theorem 3.16.

Corollary 7.5. Let $L \in 2^M$ and c be a closure operator on 2^M. If $\mathsf{O}(L) = c(L) \cap c(\overline{L})$ then $\mathsf{O}^3(L) = \mathsf{O}^2(L)$.

8. More Distinguishability Operations

A natural extension of D, as defined in Equation (4), is to consider the prefix operator and the infix operator, thus, $\mathsf{E}(L) = \mathsf{pref}(L) \cap \mathsf{pref}(\overline{L})$, and $\mathsf{F}(L) = \mathsf{infix}(L) \cap \mathsf{infix}(\overline{L})$. Because pref and infix are closure operators, E and F will share properties of D. In particular, $\mathsf{E}(L)$ is prefix-closed, $\mathsf{F}(L)$ is infix-closed, and both satisfy Corollary 7.5. If L is $\emptyset$ or $\Sigma^\star$, then $\mathsf{E}(L) = \mathsf{F}(L) = \emptyset$.

In the following subsections we briefly consider these operators.

8.1. Right Distinguishability

Given a language L, the (Myhill-Nerode) relation on $\Sigma^\star$ $x \eqsim_L y$ if $(\forall u \in \Sigma^\star)\ ux \in L \Leftrightarrow uy \in L$ is an equivalence relation and left invariant. If L is regular, the relation $\eqsim_L$ has finite index. The *right quotient* of L by a word $u \in \Sigma^\star$ is the language $Lu^{-1} = \{x \in \Sigma^\star \mid xu \in L\}$ and corresponds to an equivalence class of $\eqsim_L$, [12, 21]. For $x, y \in \Sigma^\star$, we define $\mathsf{E}_L(x, y) = L^{-1}x\Delta L^{-1}y$. Then, if we define the *right distinguishability language* of L by

$$\mathsf{E}(L) = \{w \mid \exists x, y \in \Sigma^\star\ (wx \in L\ \wedge\ wy \notin L)\}, \tag{12}$$

it is immediate that

$$\mathsf{E}(L) = \bigcup_{x \in \Sigma^\star} Lx^{-1} \setminus \bigcap_{x \in \Sigma^\star} Lx^{-1},$$

and

$$\mathsf{E}(L) = \mathsf{pref}(L) \cap \mathsf{pref}(\overline{L}).$$

For $u \in \Sigma^\star$, $(Lu^{-1})^R = (u^R)^{-1}L^R$, i.e., the right quotients of L are exactly the reversals of the (left) quotients of L^R. If L is regular, the (left) quotients of L^R correspond to the atoms of L, [6]. In this case, $\mathsf{E}(L)$ is the language of the words that distinguish between pairs of different atoms of L. We have that

$$\mathsf{E}(L) = (\mathsf{D}(L^R))^R, \tag{13}$$

i.e., $\mathsf{D}(L)^R = \mathsf{E}(L^R)$.

Lemma 8.1. Let L be a language. If L does not have $\emptyset$ as a quotient, then $\mathsf{E}(L) = \mathsf{pref}(\overline{L})$.

Proof:
Because $\emptyset$ is not a quotient of L, we have $\mathsf{pref}(L) = \Sigma^\star$, therefore $\mathsf{E}(L) = \mathsf{pref}(\overline{L})$. □

We have $\mathsf{E}(L) = L$ if and only if $\mathsf{D}(L^R) = L^R$. In particular, L has $\emptyset$ as a right quotient if and only if L^R has $\emptyset$ as quotient. The fact that L has an empty right quotient does not imply that L has an empty (left) quotient, as can be seen with $L = (a+b)^\star a$, where $\mathsf{E}(L) = \Sigma^\star$. The results in Lemma 8.2 follow immediately from 3.17–3.21.

Lemma 8.2. Let L be a language. Then the following statements hold true:

i) if L has $\emptyset$ as a right quotient, then $\mathsf{E}(L) = \mathsf{pref}(L)$.

ii) if L is prefix-closed and L has a $\emptyset$ as a right quotient, then $\mathsf{E}(L) = L$.

iii) if L is regular and $\mathsf{E}(L) = L$, then L has $\emptyset$ as a right quotient.

Example 8.3. In Figure 8 one can see, from left to right, the minimal DFA accepting the language L, the language $\mathsf{E}(L)$, $\mathsf{E}(L) \neq \mathsf{E}^2(L)$, and the language $\mathsf{E}^2(L) = \mathsf{E}^n(L)$, for $n \geq 3$.

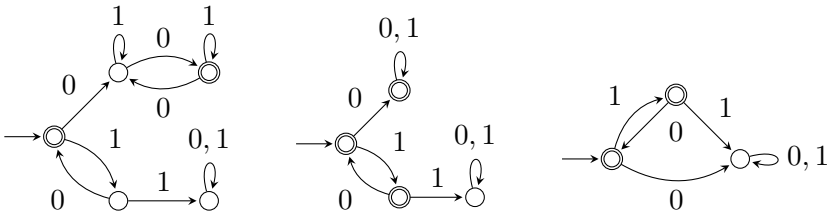

Figure 8. Automata for the languages L, $\mathsf{E}(L)$, and $\mathsf{E}^n(L)$, $n \geq 2$.

Corollary 8.4. If L is a finite language, then $\mathsf{E}(L) = \mathsf{pref}(L)$.

The state complexity of the E operation is given by the following theorem.

Theorem 8.5. If L is recognized by a minimal DFA with $n \geq 2$ states, then $sc(\mathsf{E}(L)) = n$.

Proof:
If both L and $\overline{L}$ do not have $\emptyset$ as a quotient, then $\mathsf{E}(L) = \Sigma^\star$ and only one state is needed for a DFA accepting $\mathsf{E}(L)$. Otherwise, let $\mathcal{A} = (Q, \Sigma, \delta, i, F)$ be the minimal DFA recognizing L with $|Q| = n$. We have that at least one of $\mathcal{A}$ or $\overline{\mathcal{A}}$ has a dead state. To obtain a DFA for $\mathsf{pref}(L)$ one needs only to consider all states of $\mathcal{A}$ final, except the dead state, if it exists. To get a DFA for $\mathsf{E}(L)$, we also need to exclude from the set of final states the possible dead state of the DFA $\overline{\mathcal{A}}$, recognizing $\overline{L}$, which coincides with $\mathcal{A}$, except that the set of final states is $Q \setminus F$. Tightness is achieved for the family of languages $L_n = \{a^i \mid i \leq n-2\}$, which are prefix closed, [5]. □

In Section 5, we have considered the language of the shortest words that distinguish pairs of left quotients of L, $\underline{\mathsf{D}}(L)$. In this case, we can define $\underline{\mathsf{E}}(L) = \{\underline{\mathsf{E}}_L(x,y) \mid x \not\approx_L y\}$, where $\underline{\mathsf{E}}_L(x,y) = \min\{w \mid w \in \mathsf{E}_L(x,y)\}$ if $x \not\approx_L y$, and minimum is considered with respect to the quasi-lexicographical order.

If L is regular and using Equation (13), one can have a finite set of words that distinguish between right quotients, namely $(\underline{\mathsf{D}}(L^R))^R$. However, using the notion of atoms we can compute directly $\underline{\mathsf{E}}$. As we seen before, $\mathsf{E}(L)$ distinguishes between pairs of atoms of L. To estimate the number of elements of $\underline{\mathsf{E}}(L)$, we recall the relation between atoms and right quotients.

Let $\mathcal{A} = (Q = \{0, \ldots, n-1\}, \Sigma, \delta, i, F)$ be the minimal DFA recognizing L and let R_i, $0 \leq i \leq n-1$ be the left quotients of L. Each atom can be characterized by a set $S \subseteq Q$ such that $A_S = \bigcap_{i \in S} R_i \bigcap \bigcap_{i \notin S} \overline{R_i}$. Every $x \in \Sigma^\star$ belongs exactly to one atom A_{S_x}, and if $x \eqsim_L y$, i.e, $Lx^{-1} = Ly^{-1}$, then x and y belong to the same atom. Thus, the minimal word that distinguishes two distinct right quotients with correspondent sets S and S' is

$$\min\{w \mid w \in L_S \nLeftrightarrow w \in L_{S'}\},$$

where $L_T = \bigcup_{i \in T} L_i$ for $T \subseteq Q$ and $\min\{w \mid w \in L_T\} = \min\{x_{\mathcal{A}}(i) \mid i \in T\}$. Therefore, $\underline{\mathsf{E}}_L(x, y) = \min\{x_{\mathcal{A}}(i) \mid i \in S_x \Delta S_y\}$. Using Theorem 5.6, it follows that the number of elements of $\underline{\mathsf{E}}(L)$ is less than or equal to $n-1$. We also have that $\underline{\mathsf{E}}$ is prefix closed, $\underline{\mathsf{E}}^{n+1}(L) \subseteq \underline{\mathsf{E}}^n(L)$, for all $n \geq 1$, and we can reach the fixed point in maximum $n-2$ iterations, using Theorem 5.8 and Theorem 5.9, with $W_n = \mathsf{pref}(10^n)$.

8.2. Two-sided Distinguishability

Given a language $L \subseteq \Sigma^\star$, we can define the (Myhill-Nerode) equivalence relation on $\Sigma^\star \times \Sigma^\star$, $(x, y) \not\approx_L (x', y')$ if and only if $(\forall u \in \Sigma^\star)\ xuy \in L \Leftrightarrow x'uy' \in L$. For $u, v \in \Sigma^\star$, the two-sided quotient $u^{-1}Lv^{-1} = \{x \in \Sigma^\star \mid uxv \in L\}$ corresponds to an equivalence class of $\approx_L$ and the relation $\approx_L$ is of finite index if and only if L is regular. We note that $u^{-1}Lv^{-1} = (u^{-1}L)v^{-1} = u^{-1}(Lv^{-1})$. Two-sided quotients were recently used to define biautomata, deterministic versions of which recognize exactly regular languages, [16, 17], and *couple* NFAs, which can recognize linear languages, [12].

We define the *two-sided distinguishability language* of L by

$$\mathsf{F}(L) = \{w \mid \exists x, y, x', y' \in \Sigma^\star\ (xwy \in L\ \wedge\ x'wy' \notin L)\}. \tag{14}$$

It is immediate that

$$\mathsf{F}(L) = \bigcup_{x,y \in \Sigma^\star} x^{-1}Ly^{-1} \setminus \bigcap_{x,y \in \Sigma^\star} x^{-1}Ly^{-1},$$

and

$$\mathsf{F}(L) = \mathsf{infix}(L) \cap \mathsf{infix}(\overline{L}).$$

Please note that for all languages L, $\mathsf{D}(L) \subseteq \mathsf{F}(L)$ and $\mathsf{E}(L) \subseteq \mathsf{F}(L)$. If L has $\emptyset$ as a (left) quotient, then $\emptyset$ is also a two-sided quotient. The following lemmata shows that F is always an infix operation.

Lemma 8.6. Let L be a language. If L does not have $\emptyset$ as a quotient, $\mathsf{F}(L) = \mathsf{infix}(\overline{L})$.

Proof:
Since $\emptyset$ is not a quotient of L, it follows that $\mathsf{infix}(L) = \Sigma^\star$, therefore $\mathsf{F}(L) = \mathsf{infix}(\overline{L})$. □

Lemma 8.7. Let L be a language. If L has $\emptyset$ as a quotient, then $\mathsf{F}(L) = \mathsf{infix}(L)$.

Proof:
We know that $\mathsf{suff}(\overline{L}) = \Sigma^\star$, thus $\mathsf{infix}(\overline{L}) = \Sigma^\star$. □

If L is infix closed, then L is also suffix and prefix closed. Excluding $\Sigma^\star$, the fixed points of F are exactly the infix-closed languages. To see that, by the previous lemma we have:

Lemma 8.8. If L is a infix-closed language and L has a $\emptyset$ as a quotient, then $\mathsf{F}(L) = L$.

Lemma 8.9. Let L be a language. If $\mathsf{F}(L) = L$, then L has $\emptyset$ as a quotient.

Proof:
Assume L does not have $\emptyset$ as a quotient. By Lemma 8.6, it follows $\mathsf{F}(L) = \mathsf{infix}(\overline{L})$, thus L would not be a fixed point of F. □

From these two lemmata, one has

Theorem 8.10. If L is a language different from $\Sigma^\star$, $\mathsf{F}(L) = L$ if and only if L is infix closed.

Corollary 8.11. If L is a finite language, then $\mathsf{F}(L) = \mathsf{infix}(L)$.

In case $L = \Sigma^\star$, $\mathsf{F}(L) = \Sigma^\star \cap \emptyset = \emptyset$. Because, $\mathsf{F}(\emptyset) = \emptyset \cap \Sigma^\star = \emptyset$, we have $\mathsf{F}^2(L) = \mathsf{F}(L)$. This result can be generalized for all languages L, such that $\mathsf{F}(L) \neq \Sigma^\star$.

Corollary 8.12. Given a language L, if $\mathsf{F}(L) \neq \Sigma^\star$ then $\mathsf{F}^2(L) = \mathsf{F}(L)$.

Proof:
If $\mathsf{F}(L) \neq \Sigma^\star$, then either L or $\overline{L}$ has $\emptyset$ as a quotient. Hence, $\mathsf{F}(L) = \mathsf{infix}(L)$ and $\mathsf{F}^2(L) = \mathsf{infix}(\mathsf{infix}(L)) = \mathsf{F}(L)$ or $\mathsf{F}(L) = \mathsf{infix}(\overline{L})$ and $\mathsf{F}^2(L) = \mathsf{infix}(\mathsf{F}(L)) = \mathsf{F}(L)$. □

If L is a regular language, the state complexity of F coincides with the state complexity of the infix operation.

Theorem 8.13. If L is recognized by a minimal DFA with $n \geq 2$ states, then $sc(\mathsf{F}(L)) = 2^{n-1}$.

Proof:
If both L and $\overline{L}$ do not have $\emptyset$ as a quotient, then $\mathsf{F}(L) = \Sigma^\star$, therefore only one state is needed for a DFA accepting $\mathsf{F}(L)$. Otherwise, let $\mathcal{A} = (Q, \Sigma, \delta, i, F)$ be the minimal DFA recognizing L with $|Q| = n$, hence at least one of $\mathcal{A}$ or $\overline{\mathcal{A}}$ has a dead state. An NFA recognizing $\mathsf{infix}(L)$ can be obtained by marking as initial and final all states of Q and deleting the possible dead states. The correspondent DFA has at most 2^{n-1} states, [5]. An analogous construction can be used for $\mathsf{infix}(\overline{L})$. Considering Lemma 8.6 and Lemma 8.7, a DFA for $\mathsf{F}(L)$ is one of the above. Tightness is achieved for the family of languages recognized by DFAs represented in Figure 7. □

In this case, we can also define $\underline{\mathsf{F}}(L) = \{\underline{\mathsf{F}}_L(x, y) \mid x \not\approx_L y\}$, where $\underline{\mathsf{F}}_L(x, y) = \min\{w \mid w \in \mathsf{F}_L(x, y)\}$. Although it is easy to see that $\underline{\mathsf{F}}(L)$ enjoys similar properties as $\underline{\mathsf{D}}(L)$ and $\underline{\mathsf{E}}(L)$, we leave open how to compute this set.

9. Conclusion

In this paper we have introduced language operations that help us to distinguish non-equivalent words under the Myhill-Nerode equivalence and related equivalences.

The operation D finds all the words that distinguish pairs of different left quotients of a given language. This set contains exactly the words that are simultaneously the suffixes of the language and of its complement. We show that D has a fixed point under iteration and the number of iterations is bounded by two. For a regular language L with state complexity n, we show that $\mathsf{D}(L)$ is regular with state complexity bounded by $2^n - n$. This bound is tight for Brzozowski's universal witness, U_n. We also studied the operation $\underline{\mathsf{D}}$ that produces only the minimal words that distinguish pairs of different left quotients of the language, where minimum is considered with respect to the quasi-lexicographical order. If the language is regular we show that $\underline{\mathsf{D}}$ has also a fixed point under iteration and the number of iterations until a fixed point is reached is bounded by the state complexity of the starting language. In the case of the $\underline{\mathsf{D}}$ operation, the maximum number of words in the language is $n-1$, where n is the state complexity of the original (regular) language. We used $\underline{\mathsf{D}}$ to recover the original (regular) language L as an l-cover language of an initial segment of the language, where words have length at most l, by generating words in the language up to length $l + d$, where d is the length of the longest word in $\underline{\mathsf{D}}(L)$.

We have generalized some results for these types of operations with arbitrary closures and Boolean operations. We have extended the study to infix and prefix operators to distinguish right quotients and two-sided quotients of a language.

As open problems and future work, for regular languages we can consider the state complexity of combined operations, when one of them is in the set $\{\mathsf{D}, \mathsf{E}, \mathsf{F}\}$. Is worth mentioning that recovering the whole language from a finite number of words in the language is very useful in learning algorithms, thus it would be useful to study all conditions that can help us to reconstruct it, if we know some of the distinguishability languages.

For arbitrary languages, we gave some examples and we proved only general results. We plan to address in more detail this case of non-regular languages in a future paper.

Acknowledgements

We gratefully acknowledge the constructive comments of the referees and the fruitful discussions with Rudi Freund which helped us to improve the paper.

References

[1] Bell, J., Brzozowski, J. A., Moreira, N., Reis, R.: Symmetric Groups and Quotient Complexity of Boolean Operations, *Proc. 41st ICALP* (J. Esparza, P. Fraigniaud, T. Husfeldt, E. Koutsoupias, Eds.), 8573, Springer, 2014, ISBN 978-3-662-43950-0.

[2] Blumer, A., Blumer, J., Haussler, D., Ehrenfeucht, A., Chen, M. T., Seiferas, J. I.: The Smallest Automaton Recognizing the Subwords of a Text, *Theor. Comput. Sci.*, **40**, 1985, 31–55.

[3] Brzozowski, J. A.: In Search of Most Complex Regular Languages, *Int. J. Found. Comput. Sci.*, **24**(6), 2013, 691–708.

[4] Brzozowski, J. A., Grant, E., Shallit, J.: Closures in Formal Languages and Kuratowski's Theorem, *Int. J. Found. Comput. Sci.*, **22**(2), 2011, 301–321.

[5] Brzozowski, J. A., Jirásková, G., Zou, C.: Quotient Complexity of Closed Languages, *Theory Comput. Syst.*, **54**(2), 2014, 277–292.

[6] Brzozowski, J. A., Tamm, H.: Theory of Átomata, *Proc. 15th DLT* (G. Mauri, A. Leporati, Eds.), 6795, Springer, 2011, ISBN 978-3-642-22320-4.

[7] Brzozowski, J. A., Tamm, H.: Complexity of Atoms of Regular Languages, *Int. J. Found. Comput. Sci.*, **24**(7), 2013, 1009–1028.

[8] Câmpeanu, C.: Cover Languages and Implementations, *Proc. 18th CIAA* (S. Konstantinidis, Ed.), 7982, Springer, 2013, ISBN 978-3-642-39273-3.

[9] Câmpeanu, C., Moreira, N., Reis, R.: The Distinguishability Operation on Regular Languages, *Proc. 6th NCMA 2014* (S. Bensch, R. Freund, F. Otto, Eds.), Oesterreichische Computer Gesellschaft, 2014.

[10] Câmpeanu, C., Moreira, N., Reis, R.: On the Dissimilarity Operation on Finite Languages, *Eighth Workshop on Non-Classical Models of Automata and Applications (NCMA 2016)* (H. Bordihn, R. Freund, B. Nagy, G. Vaszil, Eds.), 321, Österreichische Computer Gesellschaft, 2016.

[11] Câmpeanu, C., Păun, A.: Counting the Number of Minimal DFCA Obtained by Merging States, *Int. J. Found. Comput. Sci.*, **14**(6), 2003, 995–1006.

[12] Champarnaud, J., Dubernard, J., Jeanne, H., Mignot, L.: Two-Sided Derivatives for Regular Expressions and for Hairpin Expressions, *Proc. 7th LATA 2013* (A. H. Dediu, C. Martín-Vide, B. Truthe, Eds.), 7810, Springer, 2013.

[13] Demaine, E. D., Eisenstat, S., Shallit, J., Wilson, D. A.: Remarks on Separating Words, *Proc. 13th DCFS* (M. Holzer, M. Kutrib, G. Pighizzini, Eds.), 6808, Springer, 2011, ISBN 978-3-642-22599-4.

[14] Ginsburg, S.: On the Length of the Smallest Uniform Experiment which Distinguishes the Terminal States of a Machine, *J. ACM*, **5**(3), 1958, 266–280.

[15] Ginsburg, S., Spanier, E.: Distinguishability of a Semi-group by a Machine, *Proceedings of the American Mathematical Society*, **12**(4), 1961, 661–668.

[16] Holzer, M., Jakobi, S.: Nondeterministic Biautomata and Their Descriptional Complexity, *Proc. 15th DCFS 2013* (H. Jürgensen, R. Reis, Eds.), 8031, Springer, 2013.

[17] Klíma, O., Polák, L.: On Biautomata, *RAIRO - Theor. Inf. and Applic.*, **46**(4), 2012, 573–592.

[18] Kuratowski, C.: Sur l'Operation $\overline{A}$ de l'Analysis Situs, *Fund. Math.*, **3**, 1922, 182–199.

[19] Mohri, M., Moreno, P., Weinstein, E.: General Suffix Automaton Construction Algorithm and Space Bounds, *Theor. Comput. Sci.*, **410**(37), 2009, 3553–3562.

[20] Restivo, A., Vaglica, R.: A Graph Theoretic Approach to Automata Minimality, *Theor. Comput. Sci.*, **429**, 2012, 282–291.

[21] Sakarovitch, J.: *Elements of Automata Theory*, Cambridge University Press, 2009.

[22] Sempere, J. M.: Learning Reversible Languages with Terminal Distinguishability, *Proc. 8th ICGI 2006* (Y. Sakakibara, S. Kobayashi, K. Sato, T. Nishino, E. Tomita, Eds.), 4201, Springer, 2006, ISBN 3-540-45264-8.

[23] Yu, S.: Regular languages, in: *Handbook of Formal Languages* (G. Rozenberg, A. Salomaa, Eds.), vol. 1, Springer, 1997, 41–110.