Minimal DFAs Witnessing Language Inequivalence

Jan Martens
Leiden Institute of Advanced Computer Science, Leiden University, The Netherlands

Abstract

We study small witnesses for the inequivalence of two regular languages. A natural witness is a distinguishing word, e.g. a word in exactly one of the two languages. We propose using more succinct witnesses in the form of witnessing DFAs. A witnessing DFA recognizes a subset of one of the languages and contains at least one distinguishing word. In this way the DFA expresses behaviour contained in the first language but not the second. We show witnessing DFAs can be used to present more concise witnesses for the inequivalence of two regular languages. We show that the decision problem for the existence of a witnessing DFA of certain size is NP-complete in general, and in P in the special case of unary DFAs. Besides these computational aspects, we study structural properties of witnessing DFAs. Not all languages can be a minimal witness. It turns out that minimal witnesses are exactly the languages that are not decomposable in the union of languages with smaller state-complexity, the so-called prime languages as studied earlier by Kupferman and Mosheiff.

2012 ACM Subject Classification Theory of computation → Regular languages

Keywords and phrases Deterministic Finite Automata, Language Inequivalence, DFA decomposition, Prime languages

Digital Object Identifier 10.4230/LIPIcs.CSL.2026.44

Related Version *Based on earlier note*: `https://arxiv.org/abs/2306.03533` [13]

1 Introduction

We are motivated by computing witnesses that explain difference in behaviour of two Deterministic Finite Automata (DFAs). In the formal verification of complex systems it can be key to compute witnesses and explanations for certain properties. Rather than a computer telling two DFAs are inequivalent, we would like to compute a witness that explains why the DFAs differ. We focus on the *smallest* witnesses since simple explanations are usually the best explanations.

A straightforward approach to explain the inequivalence of two DFAs would be to provide a distinguishing word, i.e. a word that is accepted by one of the automata but not the other. The method of finding minimal length distinguishing words is well understood and decidable in polynomial time [15]. An efficient implementation is given in [19] that has the same runtime complexity as the best known algorithm for DFA minimization known as Hopcroft's minimization [8].

A natural question is if more powerful properties could result in smaller witnesses explaining the inequivalence of languages. Given two DFAs A_1 and A_2, any property of the language recognized by A_1 that is not satisfied by the language of A_2 can be used to witness their inequivalence. We propose to use inclusion of a third language as property. In this way a DFA A contains the property A_p if the language of A_p is included in the language recognized by A. In this work we study DFAs that witness inequivalence of two regular languages. In particular, a minimal witnessing DFA can be much more succinct than a distinguishing word.

The state complexity, also called *index*, of a language L is the number of states of the minimal DFA recognizing L. This descriptive complexity measure is well studied in literature, and forms a natural notion of size of a regular language. In this way properties expressed by DFAs are smaller witnesses that provide shorter and more intuitive explanation.

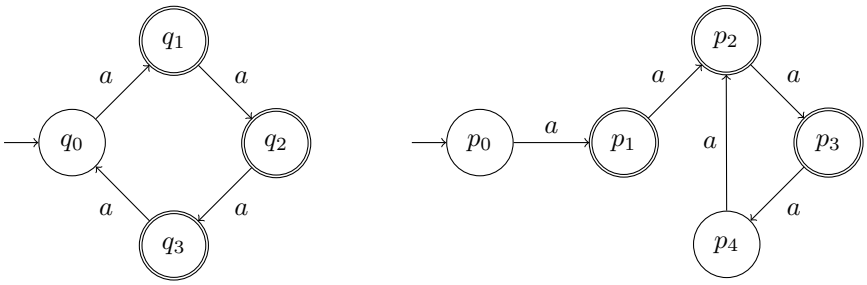

Figure 1 The DFA $\mathcal{A}$ on the left and the DFA $\mathcal{B}$ on the right side.

For example, consider the DFAs $\mathcal{A}$ and $\mathcal{B}$ shown in Figure 1. The shortest distinguishing word for these DFAs is a^7. Indeed, we confirm $a^7 \in \mathcal{L}(\mathcal{A})$ but $a^7 \notin \mathcal{L}(\mathcal{B})$. A different explanation for the inequivalence of $\mathcal{A}$ and $\mathcal{B}$ could be: every odd length sequence of a's is accepted. This property is satisfied by $\mathcal{A}$ and is not true for $\mathcal{B}$.

We call a DFA a witness for two DFAs if the language recognized is a subset of exactly one of the two DFAs. It is then said to distinguish the two languages. In the example from Figure 1, we see that the minimal witnessing DFA contains only two states, i.e. the DFA A_{odd} such that $\mathcal{L}(A_{odd}) = \{a^{2i+1} \mid i \in \mathbb{N}\}$. An automaton recognizing only the minimal distinguishing word a^7 would contain at least eight states. Therefore, a witnessing DFA can be much more succinct than a distinguishing word.

In this paper we study witnessing DFAs. We show that there are families of DFAs for which there are witnessing DFAs of constant size, where the length of the minimal distinguishing word grows linearly in the size of the DFAs. Additionally, we prove that computing minimal witnessing DFAs is NP-complete. In the case of unary DFAs this problem is computable in polynomial time.

This last result is due to a connection of witnessing DFAs with so-called prime languages [12]. Prime languages are the languages for which there is no decomposition in languages of smaller DFAs. We show that a DFA is a minimal witness for inequivalence of regular languages if and only if it is prime.

The reduction from CNF-SAT that proves the NP-completeness of deciding witnessing DFAs encodes satisfying assignment in structure of concatenation instead of composition used in other hardness proofs [11]. To show that this construction is more widely useable we show how modify the construction to obtain a simpler proof of the coNP-hardness of deciding the minimal pumping constant.

Structure

This work is structured as follows. First in Section 2 we cover the notation we use and introduce the main concept of witnessing DFAs. In Section 3 we show that computing minimal witnesses is NP-complete. After that, in Section 4, we demonstrate that the proof of the previous section can be more widely used in DFA decision problems. In Section 5 we discuss the connection between minimal witnessing DFAs, language compositionality, and so-called prime languages. In Section 6 we relate our results to the problem of separating words [4]. Finally, in Section 7 we introduce two polynomial time algorithms. The first algorithm computes a witnessing DFA that is not necessarily minimal. The second algorithm computes a minimal DFA that witnesses the non-inclusion of two unary languages. We end with conclusions and related work.

2 Notation & Preliminaries

For two natural numbers $i, j \in \mathbb{N}$ we write $[i, j] = \{i, i+1, \ldots, j\}$ as the closed interval from i to j. Given a finite alphabet Σ, a finite sequence of elements of Σ is called a word. We write ε for the empty word and define Σ^i as the set of all words over Σ of length i, and $\Sigma^* = \bigcup_{i \in \mathbb{N}} \Sigma^i$ for all words over Σ. We write $\Sigma^+ = \Sigma \cdot \Sigma^*$ for all non-empty words over Σ. Given words $u, v \in \Sigma^*$, we write $u \cdot v$ and uv for word concatenation. Additionally, given a number $i \in \mathbb{N}$ and a word $u \in \Sigma^*$ we write u^i for the concatenation of i times the word u.

▶ **Definition 1.** *A Deterministic Finite Automaton (DFA) $A = (Q, \Sigma, \delta, q_0, F)$ is a five-tuple consisting of:*

- *Q a finite set of states,*
- *Σ a finite set of symbols called the alphabet,*
- *$\delta : Q \times \Sigma \to Q$ the transition function,*
- *$q_0 \in Q$ the initial state, and*
- *$F \subseteq Q$ the set of final states.*

The transition function δ extends naturally to a transition function for words $\delta^* : Q \times \Sigma^* \to Q$. This is done inductively as follows:

$$\delta^*(q, \varepsilon) = q$$
$$\delta^*(q, aw) = \delta^*(\delta(q, a), w).$$

The language recognized by a DFA $A = (Q, \Sigma, \delta, q_0, F)$ is denoted by $\mathcal{L}(A)$, and consists of all words $w \in \Sigma^*$ such that $\delta^*(q_0, w) \in F$. A language $L \subseteq \Sigma^*$ is called *regular* iff there is a DFA A such that $\mathcal{L}(A) = L$. We write $\bar{A}$ for the DFA $\bar{A} = (Q, \Sigma, \delta, q_0, Q \setminus F)$, which recognizes the complement of A.

The *Myhill-Nerode theorem* is a useful tool to establish the number of states necessary to recognize a language. It is based on the equivalence relation between words that have the exact same accepting extensions.

▶ **Definition 2.** *Let $x, y \in \Sigma^*$ be words and $L \subseteq \Sigma^*$ a language, then $x \equiv_L y$ if and only if for all $z \in \Sigma^*$ it holds that $xz \in L \iff yz \in L$.*

▶ **Theorem 3** (Myhill-Nerode [16, 17])**.** *Let $L \subseteq \Sigma^*$ be a language, then L is regular if and only if the relation $\equiv_L$ has a finite number of equivalence classes.*

A more specific corollary of the theorem relates the number of equivalence classes of $\equiv_L$ to the smallest number of states a DFA needs in order to recognize L. We refer to this number as the index of a language and write it as $ind(L)$. For a DFA A we write $ind(A)$ for the index of the language accepted by A.

▶ **Corollary 4.** *Let L be a regular language over an alphabet Σ, then the smallest DFA A that recognizes L has $ind(L) = k$ states where k is the number of equivalence classes of the relation $\equiv_L$.*

Now we introduce the notion of witnessing DFA for the inequivalence of languages.

▶ **Definition 5.** *Given DFAs A_1, A_2 such that $\mathcal{L}(A_1) \neq \mathcal{L}(A_2)$, a DFA B is called to* witness *the language inequivalence iff:*

$$\mathcal{L}(B) \subseteq \mathcal{L}(A_1) \iff \mathcal{L}(B) \not\subseteq \mathcal{L}(A_2).$$

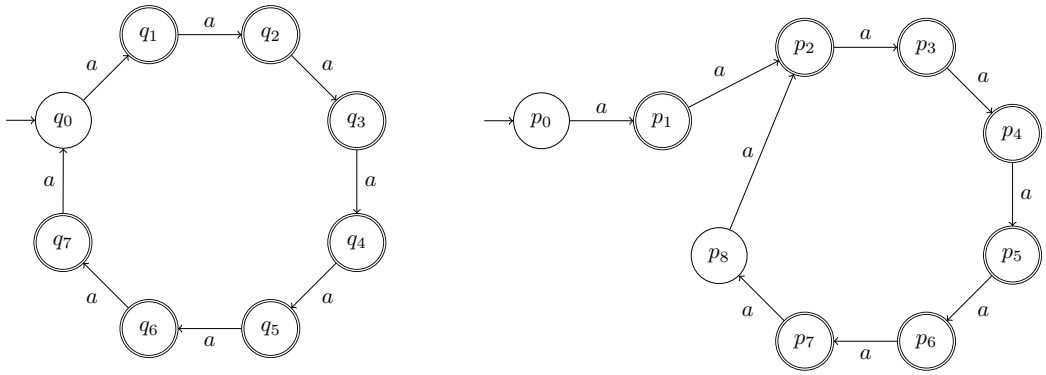

Figure 2 Example automata $\mathcal{A}_3$ (left) and $\mathcal{B}_3$ (right). The minimal witnessing word is a^{15} whereas the minimal witnessing DFA contains 2 states.

A *minimal* witness for the inequivalence of A_1 and A_2 is a witness that is minimal (in size) amongst all witnesses. Of course, for DFAs A_1 and A_2, such that $\mathcal{L}(A_1) \not\subseteq \mathcal{L}(A_2)$ the DFA A_1 itself is already a witness DFA. We are interested in small witnesses for language inequivalence, since in general small DFAs express the most intuive properties.

The notion of minimal witnessing DFAs naturally leads to the following decision problem.

k-DFA-DIST: Let A_1 and A_2 be DFAs such that $\mathcal{L}(A_1) \neq \mathcal{L}(A_2)$, and $k \in \mathbb{N}$ a number. Decide if there is a DFA A_{dist} with at most k states such that:

$$\mathcal{L}(A_{dist}) \subseteq \mathcal{L}(A_1) \iff \mathcal{L}(A_{dist}) \not\subseteq \mathcal{L}(A_2).$$

3 Minimal witnesses are NP-complete

First, we observe that DFA witnesses for language inequivalence can be arbitrarily smaller than distinguishing words. We show this by generalizing the construction from the example in the introduction (Figure 1).

▶ **Example 6.** For a number $n \in \mathbb{N} \setminus \{0\}$, let $\mathcal{A}_n$, $\mathcal{B}_n$ be DFAs defined such that:

$$\mathcal{L}(\mathcal{A}_n) = \{a^c \mid i \in \mathbb{N} \text{ and } c \neq i \cdot 2^n\}$$
$$\mathcal{L}(\mathcal{B}_n) = \{a\} \cup \{a^c \mid i \in \mathbb{N}, i \neq 0 \text{ and } c \neq 1 + i \cdot (2^n - 1)\}.$$

The minimal automata recognizing $\mathcal{L}(\mathcal{A}_n)$ (resp. $\mathcal{L}(\mathcal{B}_n)$) have 2^n (resp. $2^n + 1$) states. The smallest word accepted by $\mathcal{A}_n$ and not by $\mathcal{B}_n$ has length $2^{n+1} - 1$. In this case the DFA A_{odd} accepting all odd length sequences of a also acts as a witnessing DFA. Hence, a minimal witnessing DFA is A_{odd}, i.e. the DFA such that $\mathcal{L}(A_{odd}) = \{a^{2i+1} \mid i \in \mathbb{N}\}$. The DFA A_{odd} contains only 2 states, and thus the minimal witnessing DFA is asymptotically smaller than the minimal witnessing word.

The shape of the example automata $\mathcal{A}_3$ and $\mathcal{B}_3$ are given in Figure 2.

Next, we show the NP-completeness of computing minimal witnessing DFAs for language inequivalence. We prove k-DFA-DIST is NP-hard by a reduction from CNF-SAT.

▶ **Theorem 7.** *Deciding k-DFA-DIST is NP-complete.*

Before we introduce the reduction we define some notation in which we encode truth values of propositions. In the reduction we represent truth assignments as words over the Boolean alphabet $\mathbb{B} = \{0, 1\}$. Given a set of propositional variables $Prop = \{p_1, \ldots, p_k\}$, a truth assignment $\rho : Prop \to \mathbb{B}$ is represented by the word $a_1 \ldots a_k \in \mathbb{B}^k$, where $a_i = \rho(p_i)$ for every $i \in [1, k]$. The set $\mathcal{X} = \mathbb{B}^k$ defines all words that represent truth assignments.

Now we are ready to introduce our reduction from CNF-SAT in order to prove Theorem 7. Let $\phi = C_1 \wedge \cdots \wedge C_n$ be a CNF formula over the propositional variables $Prop = \{p_1, \ldots, p_k\}$, we define two regular languages over the alphabet $\Sigma = \mathbb{B} \cup \{\sharp\}$. The first language $L_\phi^- \subseteq \Sigma^*$ is the finite set of at most n concatenated truth assignments separated by a $\sharp$, i.e.

$$L_\phi^- = \{w_1\sharp \ldots w_j\sharp \mid j \in [1, n] \text{ and } w_1, \ldots, w_j \in \mathcal{X}\}.$$

The second language $L_\phi^+ \subseteq \Sigma^*$ is a superset of L_ϕ^-. In addition to all the words of L_ϕ^-, the language L_ϕ^+ contains words of more than n concatenated truth assignments. Specifically, L_ϕ^+ includes all words consisting of n or more concatenated truth assignments of which the first consecutively satisfy the clauses $C_1, \ldots, C_n$. More precisely,

$$L_\phi^+ = L_\phi^- \cup \{w_1\sharp \cdots w_j\sharp \mid j \geq n, w_1, \ldots w_j \in \mathcal{X} \text{ and for all } i \in [1, n] : w_i \text{ satisfies } C_i\}.$$

We sketch the shape of the languages with an example using an unsatisfiable formula.

▶ **Example 8.** Consider the CNF-formula $\mathcal{C} = p_1 \wedge \neg p_1$ over the propositional letters $Prop = \{p_1\}$. The language $L_\mathcal{C}^-$ is the language containing the words:

- $b\sharp$ for $b \in \{0, 1\}$, and
- $b\sharp b'\sharp$ for $b, b' \in \{0, 1\}$.

The language $L_\mathcal{C}^+$ contains:

- w if $w \in L_\mathcal{C}^-$, and
- $1\sharp 0\sharp w$ for every $w \in (\mathcal{X}\sharp)^+$.

Since 1 is the only truth-assignment for p_1 that satisfies $C_1 = p_1$ and only 0 satisfies $C_2 = \neg p_2$. The only words that witness the inequivalence $L_\mathcal{C}^+ \neq L_\mathcal{C}^-$ are of the shape $1\sharp 0\sharp w$, for $w \in (\mathcal{X}\sharp)^+$.

The languages L_ϕ^- and L_ϕ^+ are regular, and hence there are automata that recognize these languages. In particular, there are automata recognizing these languages that are polynomial in size. One way of observing this fact is by inspecting the number of Myhill–Nerode equivalence classes of L_ϕ^+ and L_ϕ^-.

▶ **Lemma 9.** *Given a CNF formula ϕ, the languages L_ϕ^+ and L_ϕ^- are recognizable by an automaton that is polynomial in the size of ϕ.*

The next lemma proves the key fact of our reduction. Given a CNF formula ϕ that is satisfiable on the propositional letters $Prop = \{p_1, \ldots, p_k\}$, the language $\{(w_\rho\sharp)^i \mid i \in \mathbb{N}\}$ formed by a satisfying truth assignment ρ represented as the word $w_\rho \in \{0, 1\}^k$ iterated arbitrary often is a small distinguishing automaton for the languages L_ϕ^+ and L_ϕ^-. Conversely, a distinguishing automaton smaller than a certain size necessarily implies the existence of a satisfying truth assignment for ϕ.

▶ **Lemma 10.** *Let $\phi = C_1 \wedge \cdots \wedge C_n$ be a CNF formula over k propositional letters $Prop = \{p_1, \ldots, p_k\}$. Then ϕ is satisfiable if and only if there is a DFA A_{dist} with at most $k + 2$ states such that $\mathcal{L}(A_{dist}) \subseteq L_\phi^+$ and $\mathcal{L}(A_{dist}) \not\subseteq L_\phi^-$.*

Proof. We prove both directions of the implication separately.

(⇒) Assume ϕ is satisfiable, then there is a satisfying truth assignment ρ that is mapped to the word $w_\rho = \rho(p_1)\dots\rho(p_k) \in \mathcal{X}$. We define the language $L_{dist} = \{(w_\rho \cdot \sharp)^i \mid i \in \mathbb{N}\}$, and show that L_{dist} witnesses this implication.

First we show that $L_{dist} \subseteq L_\phi^+$. Assume $i \in \mathbb{N}$, if $i \leqslant n$ then by definition $(w_\rho \cdot \sharp)^i \in L_\phi^-$ and hence also $(w_\rho \cdot \sharp)^i \in L_\phi^+$. If $i > n$, since ρ is a satisfying assignment, it holds for any $w' \in \{w\sharp \mid w \in \mathcal{X}\}^*$ that $(w_\rho \cdot \sharp)^n w' \in L_\phi^+$, and thus also $(w_\rho \cdot \sharp)^n (w_\rho \cdot \sharp)^{i-n} \in L_\phi^+$. By covering both cases this means $L_{dist} \subseteq L_\phi^+$.

Next, we observe that $(w_\rho \cdot \sharp)^{n+1} \notin L_\phi^-$, and thus $L_{dist} \not\subseteq L_\phi^-$. Hence, since $L_{dist} \subseteq L_\phi^+$ any DFA that recognizes L_{dist} is a distinguishing automaton.

The minimal DFA A_{dist} such that $\mathcal{L}(A_{dist}) = L_{dist}$ contains one loop with $k+1$ states containing all positions of the word $w_\rho \cdot \sharp$ and a sink state to reject all other words. Thus, if ϕ is satisfiable we can construct A_{dist} with $k+2$ states that distinguishes L_ϕ^+ and L_ϕ^-, which was to be showed.

(⇐) We assume A_{dist} is a DFA with at most $k+2$ states such that for the language accepted $\hat{L} = \mathcal{L}(A_{dist})$ it holds that $\hat{L} \subseteq L_\phi^+$ and $\hat{L} \not\subseteq L_\phi^-$. We show that this means ϕ is satisfiable.

Since $\hat{L} \setminus L_\phi^- \neq \emptyset$ and $L_\phi^- \subseteq L_\phi^+$ there is a word $w \in L_\phi^+ \setminus L_\phi^-$ accepted by A_{dist}. By definition w is of shape $w = w_1\sharp\dots w_n\sharp w'$ where $w' \in \Sigma^+$ and $w_1, \dots, w_n \in \mathcal{X}$ and for every $i \in [1, n]$ the word w_i represents a satisfying truth assignment for the clause C_i.

Next we show that w_1 represents a satisfying truth assignment for ϕ by counting the number of equivalence classes of $\equiv_{\hat{L}}$ for the prefixes of $w_1 \cdot \sharp$, together with the postfix $w_{post} = w_2\sharp\dots w_n\sharp w'$ that witnesses an accepting postfix for $w_1\sharp$.

We define the set U as the set containing all prefixes of $w_1 = a_1 \dots a_k$, i.e.

$$U = \{\varepsilon\} \cup \{a_1 \dots a_j \mid j \in [1, k]\}.$$

If $v, u \in U$ and $v \neq u$ then $v \not\equiv_{\hat{L}} u$, since there is a $\sigma \in \Sigma^*$ such that $v\sigma = w$ and $w \in \hat{L}$ and $u\sigma \notin \hat{L}$. This means there are $|U| = k+1$ distinct classes of $\equiv_{\hat{L}}$. Lastly, since $\sharp z \notin \hat{L}$ for any $z \in \Sigma^*$ we can also conclude that $\sharp \not\equiv_{\hat{L}} u$ for all $u \in U$.

Since we assumed that A_{dist} has at most $k+2$ states, by Corollary 4 there are at most $k+2$ equivalence classes of $\equiv_{\hat{L}}$. Since trivially $w_1\sharp \not\equiv_{\hat{L}} \sharp$, by the pigeonhole principle there is a prefix $u \in U$ such that at $w_1\sharp \equiv_{\hat{L}} u$.

It can not be the case that $u = a_1 \dots a_i$ for some $i \in [1, k]$, since

$$a_1 \dots a_i \cdot a_{i+1} \dots a_k\sharp w_{post} \in \hat{L}$$
$$w_1\sharp \cdot a_{i+1} \dots a_k\sharp w_{post} \notin \hat{L}.$$

By eliminating all alternatives we conclude $u = \varepsilon$. Using this equivalence and since $\varepsilon \cdot w_1\sharp w_{post} \in L_{dist}$ we derive that $w_1\sharp \cdot w_1\sharp w_{post} \in L_{dist}$. In particular, this means that $(w_1\sharp)^n \cdot w_{post} \in L_{dist}$. By definition of L_ϕ^+ this means that the truth assignment w_1 satisfies all clauses $C_1, \dots, C_n$ and hence it is a satisfying assignment for ϕ. This witnesses that ϕ is a satisfiable formula. ◀

This lemma allows us to prove Theorem 7.

Proof of Theorem 7. Membership of NP follows naturally. For two DFAs A_1 and A_2 we can, in polynomial time, check if $\mathcal{L}(A_1) \subseteq \mathcal{L}(A_2)$. This can be done by computing the emptiness of $\mathcal{L}(A_1) \cap \overline{\mathcal{L}(A_2)}$. Moreover, either A_1 or A_2 itself necessarily already is a distinguishing automaton, so the minimal distinguishing DFA is definitely polynomial in size.

NP-hardness is a direct consequence of Lemma 10 and of the fact that $L_\phi^- \subseteq L_\phi^+$, so the language of any distinguishing automaton is a subset of L_ϕ^+ and not vice-versa. ◀

4 Minimal Pumping Length

The pumping lemma is mostly famous for educational purposes in automata theory. The lemma asserts that in a regular language a word from a certain length can be *pumped*. A word in a language can be pumped if a non-empty part of it can be arbitrarily repeated while remaining in the language.

▶ **Definition 11** (Pumping lemma). *Let $L \subseteq \Sigma^*$ be a regular language. There is a $p \in \mathbb{N}$ such that for all words $w \in L$ if $|w| \geq p$ then there is a decomposition $w = xyz$ such that $|xy| \leqslant p$, $|y| \geq 1$ and for all $i \in \mathbb{N}$: $xy^iz \in L$.*

Recently, the computational complexity of deciding the minimal pumping length was studied in [6]. The decision problem associated with the pumping lemma is to decide if the lemma holds for a certain pumping constant p.

k-PUMPING: Let A be a DFA, and $k \in \mathbb{N}$ a number. Decide if the pumping lemma holds with constant $p = k$.

In [6] it was shown that computing the minimal pumping constant p of the lemma is coNP-hard.

▶ **Theorem 12** ([6, Cor. 15]). *k-PUMPING is coNP-complete.*

The proof of the coNP-hardness of k-pumping goes by a reduction from Hamiltonian cycle. A slightly modified language based on L_ϕ^+ from Section 3 gives an alternative proof of this fact. The reduction is directly from the tautology of DNF formulas, which is the natural coNP-complete problem.

Let the language L_ϕ^{dnf} be a language similar to the one previously mentioned, but for a propositional formula in Disjunctive Normal Form (DNF). Here, a formula ϕ is said to be in disjunctive normal formal if $\phi = C_1 \vee \cdots \vee C_n$ where each C_i is a conjunction of literals. The language contains all words $w_1\sharp \dots w_n\sharp \cdot \{w\sharp \mid w \in \mathcal{X}\}^*$, where there is an $i \in [1, n]$ such that the truth assignment w_i satisfies C_i,

$$L_\phi^{dnf} = L_\phi^- \cup \{w_1\sharp \cdots w_j\sharp \mid j \geq n, w_1, \dots, w_j \in \mathcal{X} \text{ and there is an } i \in [1, n] : w_i \text{ satisfies } C_i\}.$$

▶ **Lemma 13.** *Given a DNF formula $\phi = C_1 \vee \cdots \vee C_n$, the language L_ϕ^{dnf} is $(k+1)n$-PUMPING if and only if ϕ is a tautology.*

Proof.

(⇒) Let $w \in \mathcal{X}$ be a word representing a truth assignment. Since L_ϕ^{dnf} is $(k+1)n$-PUMPING, the pumping lemma holds for the word $(w\sharp)^n \in L_\phi^- \subseteq L_\phi^{dnf}$. Let $xyz = (w\sharp)^n$ be the composition as mentioned in the pumping lemma. Since y is non-empty, and $xy^2z \in L_\phi^{dnf}$ by the structure of L_ϕ^{dnf} it has to hold that $|y| = c(k+1)$ for some $c \in \mathbb{N}$. Observe that any such y is of the shape $w_{post}\sharp(w\sharp)^*w_{pre}$, where $w = w_{pre}w_{post}$. From this we can conclude that $xy^2z = (w\sharp)^{n+c}$ and thus, by definition of L_ϕ^{dnf}, that w represents a satisfying assignment for ϕ. In particular, since w was arbitrarily chosen, any assignment is a valid truth assignment and thus ϕ is a tautology.

(⇐) Assume ϕ is a tautology, and $w \in L_\phi^{dnf}$ such that $|w| \geq (k+1)n$. By definition the word can be represented as $w_1\sharp \dots w_j\sharp$ for some $j \geq n$ and truth assignments $w_1, \dots, w_j \in \mathcal{X}$. We distinguish on whether $|w| = (k+1)n$.

- Assume $|w| = (k+1)n$, then since ϕ is a tautology, there is a clause C_ℓ for which w_1 is a satisfying truth assignment. We verify that $w = xyz$ for the words $x = \varepsilon$, $y = w_1\sharp \dots w_{\ell-1}\sharp$ and $z = w_\ell\sharp \dots w_j\sharp$, and that $xy^iz \in L_\phi^{dnf}$ for all $i \in \mathbb{N}$.

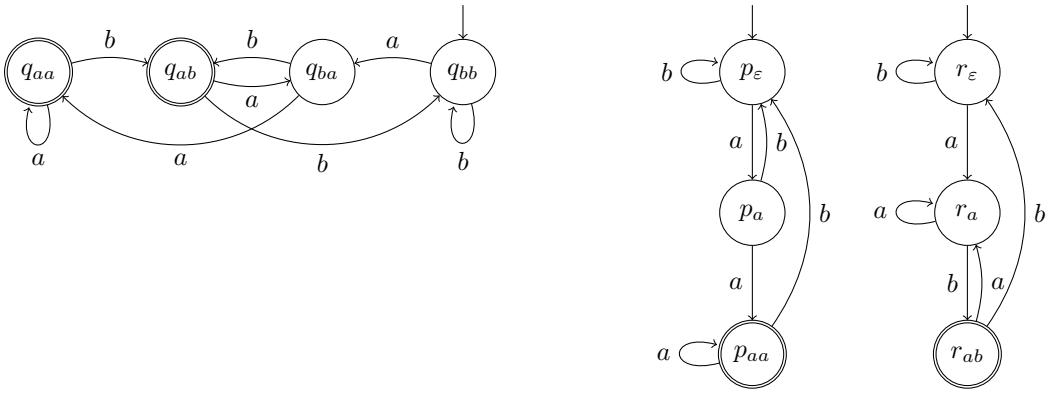

Figure 3 Minimal automata recognizing L_2 (from q_{bb}), and the languages L_a (from p_ε), and L_b (from r_ε).

- In the second case $|w| > (k+1)n$. By construction of the language $w = w_1\sharp \dots w_i\sharp$ for some $i > n$, and hence there is a word w_{n+1} that represents a truth assignment. Since ϕ is a tautology, there is a clause C_ℓ such that the word w_{n+1} represents a satisfying assignment for C_ℓ. We show that $x = w_1\sharp \dots w_{\ell-1}\sharp$, $y = w_\ell\sharp \dots w_n\sharp$, $z = w_{n+1}\sharp \dots w_j\sharp$ verifies the pumping lemma. Indeed, in xy^0z the word w_{n+1} satisfies C_ℓ and thus $xy^0z \in L_\phi^{dnf}$. For any $c \geq 1$, the word $xy^cz = w_1\sharp \dots w_n\sharp \dots w_j\sharp$, and since $w \in L_\phi^{dnf}$ also $xy^cz \in L_\phi^{dnf}$.

This shows that in both cases, the language L_ϕ^{dnf} is $(k+1)n$-PUMPING. ◀

5 Minimal witnesses are primes

In this section we study the structure of minimal witnesses and show that minimal witnesses correspond to the prime languages [12]. Prime languages are those that cannot be decomposed into a union of languages with smaller state complexity. In other words, a language L can be k-decomposed if it is the union of languages of index k. This hierarchy on regular languages was first studied in [12].

▶ **Definition 14** (k-composable). *A regular language L is k-composable if and only if there are DFAs $A_1, \dots, A_t$, such that $ind(A_i) \leqslant k$ for all $i \in [1,t]$, and*

$$L = \bigcup_{i\in[1,t]} \mathcal{L}(A_i).$$

A DFA A is called prime iff it is minimal and not k-composable for any $k < ind(A)$.

Formally in [12], instead of union decomposition, the focus is intersection decomposition. For our purpose it makes more sense to look at union decomposition. All results for intersection compositionality can be trivially dualized for union composition, this is already observed in [12]. Hence, this difference is only a matter of presentation. If a DFA A is union composite, e.g. $\mathcal{L}(A) = \bigcup_{i\in[1,n]} \mathcal{L}(A_i)$ for some DFAs $A_1, \dots, A_n$, then also $\overline{\mathcal{L}(A)} = \bigcap_{i\in[1,n]} \overline{\mathcal{L}(A_i)}$.

We show how the concept of prime languages is related to minimal witnesses. We start by the observation that not all DFAs are a minimal witness. This is illustrated by the following example.

▶ **Example 15.** Consider the language $L_2 = \{wa\sigma \mid w \in \Sigma^*, \sigma \in \Sigma\}$ over the alphabet $\Sigma = \{a, b\}$. The minimal automaton recognizing L_2 is shown in Figure 3. Observe that $ind(L_2) = 4$. Given a language $L \subseteq \Sigma^*$ such that $L_2 \not\subseteq L$, then necessarily there is a word $w \in \Sigma^*, \sigma \in \Sigma$ such that the word $wa\sigma \notin L$. Let the language L_σ be defined as $L_\sigma = \{wa\sigma \mid w \in \Sigma^*\}$. Then L_σ is a subset of L_2 and witnesses the non-inclusion $L_2 \not\subseteq L$. Since $ind(L_\sigma) = 3$, we see that for the non inclusion of L_2 in any language there is a witness of index smaller than the index of L_2.

In general for each $k \in \mathbb{N}$ define the languages $L_k = \{wau \mid w \in \Sigma^*, u \in \Sigma^k\}$. The language L_k always has a witness for non-inclusion with index exponentially smaller than the index of L_k.

A natural question is when a DFA A is a minimal witness for the non-inclusion of some languages $L \not\subseteq L'$. In the example above, the language L_2 is not prime. In fact, L_2 is 3-decomposable in the automata recognizing the languages $L_\sigma = \{wa\sigma \mid w \in \Sigma^*\}$ for each $\sigma \in \Sigma$. Now we see the language $L_2 = \bigcup_{\sigma\in\Sigma} L_\sigma$. Next, we prove that prime languages characterize exactly the languages that are minimal witnesses.

▶ **Theorem 16.** *Given a DFA A, A is a minimal witness for the non-inclusion of some DFAs B_1 in B_2, i.e. $\mathcal{L}(B_1) \not\subseteq \mathcal{L}(B_2)$ if and only if A is prime.*

Proof.

(⇐) Given a prime DFA A, we will show that there are DFAs B_1 and B_2 such that A is a minimal witness for their non-inclusion. First we define $\alpha(A)$ the *floor* of A, by

$$\alpha(A) = \{A' \mid ind(A') < ind(A) \text{ and } \mathcal{L}(A') \subseteq \mathcal{L}(A)\}.$$

Now from [12, Theorem 2.2] it holds that since A is prime, there is a *primality witness* $w \in \mathcal{L}(A)$ such that $w \notin \bigcup_{A'\in\alpha(A)} \mathcal{L}(A')$. Let $\mathcal{L}(B_2) = \Sigma^* \setminus \{w\}$. Now, for all DFAs $A' \in \alpha(A)$, in other words all DFAs such that $\mathcal{L}(A') \subseteq \mathcal{L}(A)$ and $ind(A') < ind(A)$, necessarily $w \notin \mathcal{L}(A')$. Now it is easy to see there is no smaller DFA than A that witnesses $\mathcal{L}(A) \not\subseteq \mathcal{L}(B_2)$, and hence A is a minimal witness for the non inclusion $\mathcal{L}(A) \not\subseteq \mathcal{L}(B)$.

(⇒) Assume A is a minimal witness for the DFAs B_1 and B_2. Since A is minimal, it holds for all DFAs $A' \in \alpha(A)$ that A' is no witness for the non-inclusion of B_1 and B_2. Hence, for all $A' \in \alpha(A)$ it holds that $\mathcal{L}(A') \subseteq \mathcal{L}(B_2)$. Since $\mathcal{L}(A) \not\subseteq \mathcal{L}(B_2)$ there is a primality witness $w \in \mathcal{L}(A) \setminus \mathcal{L}(B_2)$ such that $w \notin \bigcup_{A'\in\alpha(A)} \mathcal{L}(A')$, hence A is prime. ◀

A direct consequence of this theorem is that for any DFA A and $k \in \mathbb{N}$ it holds that if A is k-composable then any non-inclusion $\mathcal{L}(A) \not\subseteq \mathcal{L}(B)$ is witnessed by a DFA of index at most k. Unfortunately, the converse is not necessarily true. A witnessing DFA does not need to be part of a decomposition. For example, consider the minimal automaton A recognizing the language $\mathcal{L}(A) = \{a^{3i} \mid i \in \mathbb{N}\}$. This DFA contains 3 states and is prime. However, in order to witness that $\mathcal{L}(A) \not\subseteq \Sigma^* \setminus \{\varepsilon\}$ we can use the minimal DFA B recognizing the language $\mathcal{L}(B) = \{\varepsilon\}$, which only contains 2 states, but B is not part of the decomposition of A since A is prime.

6 Separating words

A special case of the minimal witness problem is the separating words problem. The problem is to find the smallest DFA that distinguishes two given words, in the sense that it accepts one and rejects the other. Given two words $w_1, w_2 \in \Sigma^*$ what is the minimal DFA A such that $w_1 \in \mathcal{L}(A) \iff w_2 \notin \mathcal{L}(A)$. In our context this question is exactly to compute a

witness for the inequivalence of $L_1 = \Sigma^* \setminus \{w_1\}$ and $L_2 = \Sigma^* \setminus \{w_2\}$. We define $Sep(w_1, w_2)$ as the minimal number of states in a DFA that distinguishes w_1 from w_2 – accepting one and rejecting the other.

The study of the separating words problem focuses on the asymptotic behavior of the maximum value of $Sep(w_1, w_2)$, taken over all pairs of words w_1 and w_2 of length n over an alphabet of size $\ell \in \mathbb{N}$. This is captured by the function $Sep_\ell(n)$:

$$Sep_\ell(n) = \max_{w_1, w_2 \in \{a_1, \ldots, a_k\}^n} Sep(w_1, w_2).$$

Computing $Sep(w_1, w_2)$ is NP-complete, which is a corollary from the study of identity checking in arbitrary finite semigroups [1]. Two words $w_1, w_2 \in \Sigma^*$ (often called *terms* in the context of algebras) form an *identity* for a finite semigroup S if, for every projection $\tau : \Sigma \to S$, the equality $\tau(w_1) = \tau(w_2)$ holds. Here, τ extends to words by applying the semigroup operation, that is,

$$\tau(a_1 a_2 \ldots a_n) = \tau(a_1) \cdot \tau(a_2) \cdot \ldots \cdot \tau(a_n) \quad \text{for } a_1, \ldots, a_n \in \Sigma.$$

The *identity checking* problem for a semigroup S is defined as deciding whether w_1, w_2 is an identity.

A particular semigroup is the transformation group T_n, which consists of all functions from $0, \ldots, n-1$ to itself, with function composition as operation.

▶ **Theorem 17** (([1, Thm. 2],[10, Thm. 1])). *The identity checking problem for the transformation group T_n is coNP-complete for $n \geq 3$.*

In [2] it is noticed how this is closely related to the separating words problem.

▶ **Lemma 18.** *Let $n \geq 3$ be a number. Given an alphabet Σ and words $w_1, w_2 \in \Sigma^*$. The words w_1, w_2 are not an identity in T_n if and only if there is a DFA A with at most k states such that $w_1 \in \mathcal{L}(A)$ and $w_2 \notin \mathcal{L}(A)$.*

Proof. If w_1 and w_2 are not an identity in T_n, then, by definition, there exists a projection $\tau : \Sigma \to T_n$ such that $\tau(w_1) \neq \tau(w_2)$. Note that, since $\tau(w_1) \neq \tau(w_2)$, there is an $i \in \{0, \ldots, n-1\}$ such that $\tau(w_1)(i) \neq \tau(w_2)(i)$. Given this witnessing projection τ, and number i we define an n-state DFA $A = (\{0, \ldots, n-1\}, \Sigma, \delta, q_0, F)$, where $q_0 = i$, $F = \{\tau(w_1)(i)\}$, and the transition function δ is defined as:

$$\delta(q, a) = \tau(a)(q) \quad \text{for each } q \in \{0, \ldots, n-1\} \text{ and } a \in \Sigma.$$

This DFA operates by simulating the function of the projection τ on each input symbol. By construction $w_1 \in \mathcal{L}(A)$ and since $\tau(w_2)(i) \neq \tau(w_1)(i)$, $w_2 \notin \mathcal{L}(A)$.

Conversely, assume that for words $w_1, w_2 \in \Sigma^*$ there is a DFA A with n states that distinguishes w_1 and w_2. We derive a projection $\tau : \Sigma \to T_n$ from the DFA $A = (Q, \Sigma, \delta, q_0, F)$. Without loss of generality, assume $Q = \{0, \ldots, n-1\}$ and define the projection τ for every $i \in Q, a \in \Sigma$ as $\tau(a)(i) = \delta(i, a)$. Now we observe that τ witnesses that w_1 and w_2 are not an identity in T_n. ◀

▶ **Corollary 19.** *Given an alphabet Σ, and two words $w_1, w_2 \in \Sigma^*$ deciding whether there is a DFA A with at most k states such that $w_1 \in \mathcal{L}(A)$ and $w_2 \notin \mathcal{L}(A)$ is NP-complete for any $k \geq 3$.*

It is worth noting that the function $Sep_\ell(n)$ remains equal regardless of the size of the alphabet when $\ell \geq 2$ [4].

▶ **Lemma 20** ([4, Proposition 2.]). *For all $\ell \geq 2$, $Sep_\ell(n) = Sep_2(n)$.*

However, for any finite k, the existence of a k state DFA that distinguishes two binary words can be decided in constant time by trying all projections $\tau : \{0,1\} \to T_k$. When the alphabet is part of the input and $k \geq 2$, then trying all the projections is not feasible, since there are $|T_k|^{|\Sigma|}$ possible projections.

So, deciding whether $Sep(w_1, w_2) = k$ is known to be NP-complete, for any fixed k and variable alphabet size. The complexity remains unresolved when k is part of the input and the alphabet is constant. This is particularly significant because the binary case fully determines the asymptotic behaviour of $Sep_\ell(n)$ – making it an interesting question for future research.

▶ **Question 21.** *Given $w_1, w_2 \in \{0,1\}^*$ is computation of $Sep(w_1, w_2)$ possible in polynomial time?*

7 Computing Witnesses in polynomial time

In this section we provide polynomial algorithms that compute not necessarily minimal witnesses for DFA inequivalence. First we compute witnessing DFAs greedily combining states until a fixed point is reached. Secondly, for unary DFAs we build a polynomial algorithm that computes a witness of index k or rejects if it does not exist. This algorithm uses the notion of *clean* quotients, which are used to show that deciding primality of unary DFAs is decidable in LOGSPACE [9].

Simple witnesses

In this section we introduce a naive polynomial time algorithm that computes a witnessing DFA that is not minimal, but can not become smaller by state merging.

▶ **Definition 22** (Irreducible witness). *Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA such that $\mathcal{L}(A) \not\subseteq \mathcal{L}(B)$ for some DFA B. A DFA A_{dist} is called an* irreducible witness *w.r.t. A and B iff there is no DFA $A' \in \alpha(A_{dist})$ such that $\mathcal{L}(A') \not\subseteq \mathcal{L}(B)$, where $\alpha(A_{dist})$ is the floor of A_{dist} defined as*

$$\alpha(A_{dist}) = \{A' \mid ind(A') < ind(A_{dist}) \text{ and } \mathcal{L}(A') \subseteq \mathcal{L}(A_{dist})\}.$$

We obtain irreducible DFAs by iteratively combining states. In order to do so, we define an operation on DFAs that maps one state to another. Given a DFA $A = (Q, \Sigma, \delta, q_0, F)$, and two states $q, p \in Q$, when we write $A[q \mapsto p]$, we mean the automaton where the transitions into q are relayed to p, i.e.

$$A[q \mapsto p] = (Q \setminus \{q\}, \Sigma, \delta', q_0', F), \text{where for all } r, a \in Q \times \Sigma$$

$$\delta'(r,a) = \begin{cases} p & \text{if } \delta(r,a) = q \\ \delta(r,a) & \text{otherwise.} \end{cases}$$

$$q_0' = \begin{cases} p & \text{if } q_0 = q \\ q_0 & \text{otherwise.} \end{cases}$$

The algorithm follows the following procedure.

1. **Pick a distinguishing word.** First we compute a distinguishing word $w \in \mathcal{L}(A_1) \setminus \mathcal{L}(A_2)$. This can be done in quasi-linear time [19]. It can be noted that a shortest distinguishing word does not necessarily result in the smallest witnessing automaton. However, by lack of better heuristics it seems best to pick a smallest distinguishing word w, since this will guarantee an upper bound on the distinguishing DFA found.

2. **Greedily combine states.** We compute the path w takes in the automaton A_1. We try to combine states on this path, and check whether the DFA still accepts a subset of A_1. By construction the DFA still accepts w and thus is a witnessing DFA for $\mathcal{L}(A_1) \not\subseteq \mathcal{L}(A_2)$.

Algorithm 1 Computing a irreducible distinguishing DFA.

Input: DFAs $A_1 = (Q_1, \Sigma, \delta_1, q_0^1, F_1)$ and $A_2 = (Q_2, \Sigma, \delta_2, q_0^2, F_2)$ such that $\mathcal{L}(A_1) \not\subseteq \mathcal{L}(A_2)$
Output: A witnessing DFA A_{dist}.

1: **function** SimpleWitness(A_1, A_2)
2: Compute a word $w \in \mathcal{L}(A_1) \setminus \mathcal{L}(A_2)$
3: $A_{dist} = (Q := Q_1 \cup \{q_{sink}\}, \Sigma, \delta_1, q_0^1, \{\delta^*(q_0^1, w)\})$
4: $Q_{visit} = \{q_0^1\}$
5: **for** $i \in [1, |w|]$ **do**
6: $Q_{visit} := Q_{visit} \cup \{\delta_1^*(q_0^1, w[1, i])\}$
7: **for** $q \in Q \setminus Q_{visit}$ **do**
8: $A_{dist} := A_{dist}[q \mapsto q_{sink}]$ ▷ Remove states not necessary to accept w
9: **while** Candidates(A_{dist}, A_1, w) $\neq \emptyset$ **do**
10: pick $(q, q') \in$ Candidates(A_{dist}, A_1, w)
11: $A_{dist} := A_{dist}[q' \mapsto q]$
return A_{dist}
12: **function** Candidates($A_{dist} = (Q, \Sigma, \delta, q_0, F), A_1 = (Q_1, \Sigma_1, \delta_1, q_0^1, F_1), w$)
13: $C := \{(q, q') \in Q \times Q \mid w \in A_{dist}[q' \mapsto q]\}$
14: **for** $q, q' \in C$ **do**
15: **if** $\mathcal{L}(A_{dist}[q' \mapsto q]) \not\subseteq \mathcal{L}(A_1)$ **then**
16: $C := C \setminus \{(q, q')\}$
return C

This procedure results in a DFA which is not larger than that of a minimal distinguishing word, and potentially is a minimal witnessing DFA. We prove that the resulting DFA is so-called *irreducible*, which is, for any smaller DFA A', which is a subset of A_1 it holds that $w \notin \mathcal{L}(A')$.

▶ **Theorem 23.** *Given two DFAs A_1 and A_2, such that $\mathcal{L}(A_1) \not\subseteq \mathcal{L}(A_2)$, and a word $w \in \mathcal{L}(A_1) \setminus \mathcal{L}(A_2)$ the output DFA* SimpleWitness(A_1, A_2) *from Algorithm 1 is irreducible.*

An interesting question is whether we can compute the minimal witnessing DFA if we compute the correct word w in Line 2. In other words, given DFA A can we compute a minimal DFA which recognizes a subset of A while still containing a word $w \in \mathcal{L}(A)$. More formally we are interested in the computational complexity of the following decision problem.

w, k-DFA-DIST: Given a DFA A_1, a number $k \in \mathbb{N}$, and a word $w \in \mathcal{L}(A_1)$ is there a DFA A_d of index k such that $\mathcal{L}(A_d) \subseteq \mathcal{L}(A_1)$ and $w \in \mathcal{L}(A_d)$.

If w, k-DFA-DIST is in P, then also Question 21 is answered affirmitively, e.g. we can compute the smallest DFA that separates two words in polynomial time.

Unary DFAs

When the alphabet Σ of the input DFAs is only a singleton $|\Sigma| = 1$, the problem of deciding minimal witnesses becomes easier. This is due to the specific shapes of these automata.

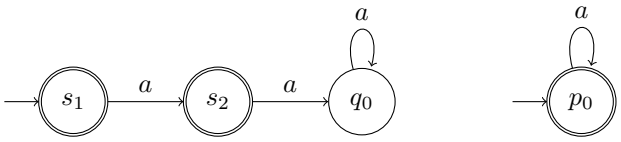

Figure 4 The $(2,1)$-DFA A_1 on the left and the $(0,1)$-DFA A_2 on the right side.

A unary DFA is a DFA where the alphabet Σ is a singleton, usually we pick $\Sigma = \{a\}$. The language of unary DFA is a set of words in the form a^i for some natural number $i \in \mathbb{N}$. It is natural to see the language of a unary DFA as a subset of $\mathbb{N}$. For a unary DFA B we define the unary-language is $\mathcal{L}_{\mathbb{N}}(B) = \{i \in \mathbb{N} \mid a^i \in \mathcal{L}(B)\}$.

A minimal unary DFA has a specific shape. Since each state has exactly one outgoing transition, it contains exactly one cycle. The initial state might not be on the cycle. In this case the state is the start of a path leading up to the cycle. For the analysis of unary DFAs it is very convenient to distinguish these states.

For two integers $d \geq 1$ and $\ell \geq 0$ a unary (ℓ, d)-DFA is a unary DFA with ℓ prefix states $s_0, \ldots, s_{\ell-1}$ and a cycle containing d states $q_0, \ldots, q_{d-1}$. The path leading into the cycle we call the prefix. Examples of these automata are given in Figure 4.The language of a unary (ℓ, d)-DFA $A = (\{s_0, \ldots s_{\ell-1}\} \cup \{q_0, \ldots q_{d-1}\}, \{a\}, \delta, s_1, F)$ is given by

$$\mathcal{L}_{\mathbb{N}}(A) = \{i \in \mathbb{N} \mid i < \ell \text{ and } s_i \in F\} \cup \{\ell + i + cd \mid c \in \mathbb{N}, 0 \leqslant i < d \text{ and } q_i \in F\}.$$

For unary DFAs computing minimal DFA witnesses is similar to deciding primality of unary DFAs. Deciding primality for unary DFAs is computable in deterministic logarithmic space [9]. An algorithm is proposed that carefully inspects so-called *clean quotients*. A clean quotient of a unary DFA is a unary DFA in which the cycle is folded into a smaller cycle.

Although our setting differs from primality testing, a similar technique can be used to compute minimal witnesses for non-inclusion. We inspect all possible covers of the cycle of the unary DFA. For each possible folding this procedure takes only polynomial time and there is only a linear number of possible covers.

In particular there is a unary DFA with a single accepting state that witnesses this non-inclusion. Hence, finding minimal witnesses for the language non-inclusion of two unary DFAs boils down to finding the smallest numbers $k_1, k_2 \in \mathbb{N}$ such that the language $L_{k_1,k_2} = \{a^{k_1} \cdot a^{i \cdot k_2} \mid i \in \mathbb{N}\}$ distinguishes A_1 and A_2.

We define the family of unary DFAs $A_{\ell,d} = (Q, \{a\}, \delta, s_0, F)$ for each $\ell, d \in \mathbb{N}$, where

- the set of states is defined as:

$$Q = \{s_0, \ldots, s_{\ell-1}\} \cup \{q_0, \ldots, q_{d-1}\},$$

- the transition function is given as:

$$\delta(p,a) = \begin{cases} s_{j+1} & \text{if } p = s_j, \text{ for some } j \in \{0, \ldots, \ell-1\} \\ q_0 & \text{if } p = s_\ell \\ q_j & \text{otherwise, if } p = q_i \text{ and } j \equiv i+1 \mod d \end{cases},$$

- and, one accepting state at the start of the cycle $F = \{q_0\}$.

Observe that there is always a minimal distinguishing DFA in the shape of $A_{\ell,d}$.

▶ **Lemma 24.** *Given unary DFAs A_1, A_2 such that $\mathcal{L}(A_1) \not\subseteq \mathcal{L}(A_2)$, then if a unary DFA B witnesses the non-inclusion, then there is a DFA $A_{\ell,d}$ such that $ind(A_{\ell,d}) \leqslant ind(B)$ and $A_{\ell,d}$ witnesses $\mathcal{L}(A_1) \not\subseteq \mathcal{L}(A_2)$.*

Proof. Let $B = (Q, \{a\}, \delta, q_0, F)$ be a distinguishing DFA such that $\mathcal{L}(B) \subseteq \mathcal{L}(A_1)$ and $\mathcal{L}(B) \not\subseteq \mathcal{L}(A_2)$. Then, by definition there is a word $w \in \mathcal{L}(B)$ such that $w \notin \mathcal{L}(A_2)$. Now we define the DFA $B' = (Q, \{a\}, \delta, q_0, \{\delta^*(q_0, w)\})$ such that the only accepting state is the one accepting w. Since B' still accepts w, it is still a distinguishing DFA for $\mathcal{L}(A_1) \not\subseteq \mathcal{L}(A_2)$ and by construction $ind(B') \leqslant ind(B)$.

We complete the proof by showing that $\mathcal{L}(B') = \mathcal{L}(A_{\ell,d})$ for some $\ell, d \in \mathbb{N}$. If $\delta^*(w, q_0)$ is part of the prefix $s_i \in Q$, then the DFA $\mathcal{L}(A_{|w|,0}) = \mathcal{L}(B')$. In the other case when $\delta^*(q_0, w) = q_i$ is some state $q_i \in Q$ on a cycle, and let that cycle have length d. Then it holds that $\mathcal{L}(A_{|w|,d}) = \mathcal{L}(B')$. ◀

Algorithm 2 Computing a minimal witnessing DFA for two unary DFAs.

Input: Unary DFAs A_1 and A_2 such that $\mathcal{L}(A_1) \not\subseteq \mathcal{L}(A_2)$
Output: A minimal witnessing DFA A_{dist}.

```
1: function UnaryMinWitness(A_1, A_2)
2:     A_min := A_1
3:     for x ∈ L_N(A_1) such that x ⩽ ind(A_1) do
4:         for 0 ⩽ d ⩽ ind(A_1) do
5:             while x > d and x − d ∈ L_N(A_1) do
6:                 x := x − d
7:             if ind(A_{x,d}) < ind(A_min) and L_N(A_{x,d}) ⊈ L_N(A_2) then
8:                 A_min := A_{x,d} where L_N(A_{x,d}) = L_{x,d}
9:     return A_min
```

There are only polynomially many of these automata of smaller index, e.g. $ind(A_{\ell,d}) \leqslant k$. A polynomial time algorithm can simply enumerate them all. Our algorithm is slightly less naive and checks for each accepting state that accepts a distinguishing word, the possible languages $A_{\ell,d}$ that accept that distinguishing word.

▶ **Lemma 25.** *Let A_1, A_2 be two unary DFAs such that $\mathcal{L}(A_1) \not\subseteq \mathcal{L}(A_2)$. Then the DFA $B =$ UnaryMinWitness(A_1, A_2), by Algorithm 2 is a minimal witness for the non-inclusion.*

Since all loops of Algorithm 2 have a linear bound, the algorithm runs in polynomial time.

▶ **Theorem 26.** *The decision problem k-DFA-DIST for unary DFAs is in P.*

8 Related work

There are some decision problems on DFAs that show some similarities, but are different from the work here. For instance the early work of Gold [5] and Pfleeger [18] in which it is shown that learning minimal DFAs from (partial) observations is NP-complete. In this line of work by Gold, so-called *separating* languages are widely studied in the literature. Here the separating problem is, given languages L_1 and L_2, to find a separating language L_{sep} such that $L_{sep} \subseteq L_1$ and $L_{sep} \cap L_2 = \emptyset$. Although this resembles our witnessing problem, a direct relation is not obvious.

The framework of distinguishing words roughly corresponds to that of Hennessy–Milner theorems for non-deterministic structures [7]. Two non-deterministic structures are not *bisimilar* if and only if there is a distinguishing formula (within the Hennessy–Milner logic). These formulas are used as counter-examples in the field of model checking. Despite the fact

that computing minimal formulas is NP-hard, a similar method as computing distinguishing words results in succinct witnesses [3, 14]. This work can contribute towards witnesses with invariants in the non-deterministic setting.

9 Conclusions

We showed how to use DFAs to witness language inequivalence. In this way it is possible more concisely represent witnesses that explain why the language of input DFAs are not equivalent. These witnesses correspond with prime languages that are previously studied [12, 9]. We show with a reduction from CNF-SAT that deciding minimal distinguishing witnesses is NP-complete. This reduction exploits structure in the language in a non-trivial way. This structure also provides a simpler proof of the coNP-hardness of the minimal pumping length. This indicates that it might be a useful tool in the DFA workbench. In particular, we conjecture that it can be used to closen the complexity gap of deciding primality [12].

To contrast this NP-hardness result we provide a greedy algorithm that in polynomial time computes a witnessing DFA that is never larger than a distinguishing word. In addition, we show that deciding minimal witnesses is in P for unary language non-inclusion.

We leave open two complexity questions that seem interesting. First the complexity of deciding w, k-DFA-DIST and the separating words problem for a binary alphabet. Additionally, it would be interesting to extend the hardness result primality of DFAs. We sketched a connection between prime DFA and minimal witnesses for DFA inequivalence, however, translating this hardness result seems not trivial.

References

1 J. Almeida, M. V. Volkov, and S. V. Goldberg. Complexity of the identity checking problem for finite semigroups. *Journal of Mathematical Sciences*, 158(5):605–614, 2009. doi:10.1007/s10958-009-9397-z.

2 Andrei A. Bulatov, Olga Karpova, Arseny M. Shur, and Konstantin Startsev. Lower bounds on words separation: Are there short identities in transformation semigroups?, August 2017. doi:10.37236/6450.

3 Rance Cleaveland. On automatically explaining bisimulation inequivalence. In Edmund M. Clarke and Robert P. Kurshan, editors, *Proc. Computer-Aided Verification (CAV 1990)*, pages 364–372, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg. doi:10.1007/BFb0023750.

4 Erik D Demaine, Sarah Eisenstat, Jeffrey Shallit, and David A Wilson. Remarks on separating words. In Markus Holzer, Martin Kutrib, and Giovanni Pighizzini, editors, *Proc. of DCFS 2011*, volume 6808 of *LNCS*, pages 147–157. Springer, 2011. doi:10.1007/978-3-642-22600-7_12.

5 Mark E. Gold. Complexity of automaton identification from given data. *Information and Control*, 37(3):302–320, 1978. doi:10.1016/S0019-9958(78)90562-4.

6 Hermann Gruber, Markus Holzer, and Christian Rauch. The pumping lemma for regular languages is hard. In *International Conference on Implementation and Application of Automata*, pages 128–140. Springer, 2023. doi:10.1007/978-3-031-40247-0_9.

7 Matthew Hennessy and Robin Milner. On observing nondeterminism and concurrency. In Jaco de Bakker and Jan van Leeuwen, editors, *Automata, Languages and Programming (ICALP '1980)*, pages 299–309, Berlin, Heidelberg, 1980. Springer-Verlag. doi:10.5555/646234.758793.

8 John Hopcroft. An n log n algorithm for minimizing states in a finite automaton. In Zvi Kohavi and Azaria Paz, editors, *Theory of Machines and Computations*, pages 189–196. Academic Press, 1971. doi:10.1016/B978-0-12-417750-5.50022-1.

9 Ismaël Jecker, Orna Kupferman, and Nicolas Mazzocchi. Unary Prime Languages. In Javier Esparza and Daniel Král', editors, *Proc. of MFCS 2020*, volume 170 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 51:1–51:12, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.MFCS.2020.51.

10 Ondřej Klíma. Identity checking problem for transformation monoids. *Semigroup Forum*, 84(3):487–498, 2012. doi:10.1007/s00233-012-9401-7.

11 Dexter Kozen. Lower bounds for natural proof systems. In *Proc. of SFCS 1977*, pages 254–266. IEEE, 1977. doi:10.1109/SFCS.1977.16.

12 Orna Kupferman and Jonathan Mosheiff. Prime languages. *Information and Computation*, 240:90–107, 2015. doi:10.1016/j.ic.2014.09.010.

13 Jan Martens. Deciding minimal distinguishing DFAs is NP-complete. *arXiv preprint arXiv:2306.03533*, 2023. doi:10.48550/arXiv.2306.03533.

14 Jan Martens and Jan Friso Groote. Computing minimal distinguishing Hennessy-Milner formulas is NP-hard, but variants are tractable. In G.A. Pérez and J.-F. Raskin, editors, *Proc. of CONCUR 2023*, volume 279 of *LIPIcs*, pages 32:1–32:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.CONCUR.2023.32.

15 Tyler Moore. Gedanken-experiments on sequential machines. In C. E. Shannon and J. McCarthy, editors, *Automata Studies, Annals of Mathematical Studies, no. 34*. Citeseer, 1956.

16 John R. Myhill. Finite automata and the representation of events. *WADD Technical Report*, 57-624:112–137, 1957.

17 Anil Nerode. Linear automaton transformations. *Proceedings of the American Mathematical Society*, 9(4):541–544, 1958. doi:10.1090/S0002-9939-1958-0135681-9.

18 Charles P. Pfleeger. State reduction in incompletely specified finite-state machines. *IEEE Transactions on Computers*, 100(12):1099–1102, 1973. doi:10.1109/T-C.1973.223655.

19 Rick Smetsers, Joshua Moerman, and David N Jansen. Minimal separating sequences for all pairs of states. In Adrian-Horia Dediu, Jan Janoušek, Carlos Martín-Vide, and Bianca Truthe, editors, *Proc. of (LATA 2016)*, volume 9618 of *LNCS*, pages 181–193. Springer, 2016. doi:10.1007/978-3-319-30000-9_14.