Logical Methods in Computer Science
Volume 22, Issue 2, 2026, pp. 16:1–16:31
https://lmcs.episciences.org/

Submitted May 20, 2025
Published May 07, 2026

DETERMINISTIC SUFFIX-READING AUTOMATA

R KEERTHAN [a], B SRIVATHSAN [b], R VENKATESH [c], AND SAGAR VERMA [c]

[a] Tata Consultancy Services Innovation Labs, Pune, India & Chennai Mathematical Institute, India
e-mail address: keerthan@cmi.ac.in

[b] Chennai Mathematical Institute, India & CNRS IRL 2000, ReLaX, Chennai, India
e-mail address: sri@cmi.ac.in

[c] Tata Consultancy Services Innovation Labs, Pune, India
e-mail address: venkatesh.rv@gmail.com, verma.sagar2@tcs.com

Abstract. We introduce deterministic suffix-reading automata (DSA), a new automaton model over finite words. Transitions in a DSA are labeled with words. From a state, a DSA triggers an outgoing transition on seeing a word *ending* with the transition's label. Therefore, rather than moving along an input word letter by letter, a DSA can jump along blocks of letters, with each block ending in a suitable suffix. This feature allows DSAs to recognize regular languages more concisely, compared to DFAs. In this work, we focus on questions around finding a "minimal" DSA for a regular language. In this context, the number of states is not a faithful measure of the size of a DSA, since the transition-labels contain strings of arbitrary length. Hence, we consider total-size (number of states + number of edges + total length of transition-labels) as the size measure of DSAs.

We start by formally defining the model and providing a DSA-to-DFA conversion that allows to compare the expressiveness and succinctness of DSA with related automata models. Our main technical contribution is a method to *derive* DSAs from a given DFA: a DFA-to-DSA conversion. A surprising observation is that the smallest DSA derived from the canonical DFA of a regular language L need not be a minimal DSA for L. This observation leads to a fundamental bottleneck in deriving a minimal DSA for a regular language. In fact, we prove that given a DFA and a number $k \geq 0$, the problem of deciding if there exists an equivalent DSA of total-size $\leq k$ is NP-complete. On the other hand, we impose a restriction on the DSA model and show that our derivation procedure can be adapted to produce a minimal automaton within this class of restricted DSAs, starting from the canonical DFA.

1. Introduction

Deterministic Finite Automata (DFA) are fundamental to many areas in Computer Science. Apart from being the cornerstone in the study of regular languages, automata have been applied in several contexts: such as text processing [MMW09], model-checking [CGK$^+$18], software verification [BJNT00, BHV04, GH01], and formal specification languages [Har87]. The popularity of automata has also led to the development of several automata handling libraries [LMS22].

A central challenge in the application of automata, in almost all of these contexts, is the large size of the DFA involved. Although automata are an intuitive way to describe computations using states and transitions, their descriptions are at a low level, resulting

DOI:10.46298/LMCS-22(2:16)2026

in automata of large size for non-trivial applications. Our objective in this paper is to investigate an automaton model with a more concise representation of the computation. Non-determinism is one way that is well-known to give exponential succinctness. However, a deterministic model is useful in formal specifications and automata implementations. The literature offers several ways to tackle the problem of large DFA size, while staying within the realm of determinism. In this work, we propose a new solution to this problem.

The size of a DFA could be large due to various reasons. One of them is simply the size of the alphabet. For instance, consider the alphabet of all ASCII characters. Having a transition for each letter from each state blows up the size of the automata. *Symbolic automata* [VHL$^+$12, DV17] have been proposed to handle large alphabets. Letters on the edges are replaced by formulas, which club together several transitions between a pair of states into one symbolic transition. If the domain is the set of natural numbers, a transition $q \xrightarrow{\text{odd(x)}} q'$ with a predicate odd(x) is a placeholder for all odd numbers [D'A]. Symbolic automata have been implemented in many tools and have been widely applied (see [D'A] for a list of tools and applications).

In text processing, automata are used as an index for a large set U of strings. Given a text, the goal is to find if it contains one of the strings from the set. A naïve DFA that recognizes the set U is typically large. *Suffix automata* (different from our model of *suffix-reading* automata) are DFA that accept the set of all suffixes of words present in U. These automata are used as more succinct indices to represent U and also make the pattern matching problem more efficient [MMW09].

Another dimension for reducing the DFA size is to consider transitions over a block of letters. *Generalized automata* (GA) are extensions of non-deterministic finite automata (NFAs) that can contain strings, instead of letters, on transitions. A word w is accepted if it can be broken down as $w_1 w_2 \dots w_k$ such that each segment is read by a transition. This model was defined by Eilenberg [Eil74], and later Hashiguchi [Has91] proved that for every regular language L there is a minimal GA in which the edge labels have length at most a polynomial function in m, where m is the size of the syntactic monoid of L.

Giammarresi *et al.* [GM99] consider *Deterministic Generalized Automata* (DGA) and propose an algorithm to generate a minimal DGA (in terms of the number of states) in which the edges have labels of length at most the size of the minimal DFA. The algorithm uses a method to suppress states of a DFA and create longer labels. The key observation is that minimal DGAs can be derived from the canonical DFA by suppressing states. The paper also throws open a more natural question of minimality in the total-size of the automaton, which includes the number of states, edges and the sum of label lengths.

A natural extension from strings on transitions to more complex objects is to consider regular expressions. The resulting automaton model is called *Expression automata* in [HW04]. Expression Automata were already considered in [BM63] to convert automata to regular expressions. Every DFA can be converted to a two state expression automaton with a regular expression connecting them. A model of *Deterministic Expression automata* (DEA) was proposed in [HW04]. To get determinism, every pair of outgoing transitions from a state is required to have disjoint regular languages, and moreover, each expression in a transition needs to be prefix-free: for each w in the language, no proper prefix of w is present in the language. This restriction makes deterministic expression automata less expressive than DFAs: it is shown that DEA languages are prefix-free regular languages. An algorithm to convert a DFA to a DEA, by repeated state elimination, is proposed in [HW04]. The resulting

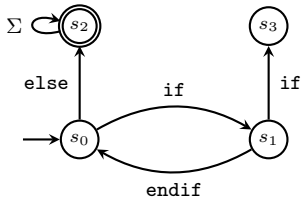

Figure 1: DSA for out-of-context `else`

DEA is minimal in the number of states. The issue with generic Expression automata is the high expressivity of the transition condition, that makes states almost irrelevant. On the other hand, DEA have restrictions on the syntax that make the model less expressive than DFAs. We consider an intermediate model.

Our model. In this work, we introduce *Deterministic Suffix-reading Automata (DSAs)*. We continue to work with strings on transitions, as in DGA. However, the meaning of transitions is different. A transition $q \xrightarrow{abba} q'$ is enabled if at q, a word w *ending* with *abba* is seen, and moreover no other transition out of q is enabled at any prefix of w. Intuitively, the automaton tracks a finite set of pattern strings at each state. It stays in a state until one of them appears as the *suffix* of the word read so far, and then makes the appropriate transition. We start with a motivating example. Consider a model for out-of-context `else` statements, in relation to `if` and `endif` statements in a programming language. Assume a suitable alphabet Σ of characters. Let $L_{\texttt{else}}$ be the set of all strings over the alphabet where (1) there are no nested `if` statements, and (2) there is an `else` which is not between an `if` and an `endif`. A DFA for this language performs string matching to detect the `if`, `else` and `endif`. The DSA is shown in Figure 1: at s_0, it passively reads letters until it first sees an `if` or an `else`. If it is an `if`, the automaton transitions to s_1. For instance, on a word `abf4fgif` the automaton goes to s_1, since it ends with `if` and there is no `else` seen so far. Similarly, at s_1 it waits for one of the patterns `if` or an `endif`. If it is the former, it goes to s_3 and rejects, otherwise it moves to s_0, and so on. (Note: in case of multiple matches, the longest match is selected. This avoids non-determinism when reading `endif`)

Suffix-reading automata have the ability to wait at a state, reading long words until a matching pattern is seen. This results in an arguably more readable specification for languages which are "pattern-intensive". This representation is orthogonal to the approaches considered so far. Symbolic automata club together transitions between a pair of states, whereas DSA can do this clubbing across several states and transitions. DGA have this facility of clubbing across states, but they cannot ignore intermediate letters, which results in extra states and transitions.

Overview of results. We formally present deterministic suffix-reading automata and its semantics, quantify its size in comparison to an equivalent DFA, and study an algorithm to construct a DSA starting from a DFA. This is in the same spirit as in DGAs, where smaller DGAs are obtained by suppressing states. For automata models with strings on transitions, the number of states is not a faithful measure of the size of a DSA. As described in [GM99] for DGAs, we consider the total-size of a DSA which includes the number of states, edges, and the sum of label lengths. The key contributions of this paper are:

Introduction to the DSA model. We formally present deterministic suffix-reading automata (Section 3), and compare its size to DFAs and DGAs (Section 4). We prove that DSAs accept regular languages, and nothing more. Every complete DFA can be seen as a DSA. For the converse, we prove that for every DSA of size k, there is a DFA with size at most $2k \cdot (1 + 2|\Sigma|)$, where Σ is the alphabet (Lemma 4.3, Theorem 4.5). This answers the question of how small DSAs can be in comparison to DFAs for a certain language : if n is the size of the minimal DFA for a language L, minimal DSAs for L cannot be smaller than $\frac{n}{2\cdot(1+2|\Sigma|)}$. When the alphabet is large, one could expect smaller sized DSAs. We describe a family of languages L_n, with alphabet size n, for which the size of the minimal DFA is quadratic in n, whereas size of the corresponding DSA is a linear function of n (Lemma 4.4).

Complexity of minimization. We focus on the minimization problem for DSAs in this paper. Minimization for DFAs can be done efficiently in polynomial-time, thanks to the Myhill-Nerode characterization. For DGAs, [GM99] provides a polynomial-time method to find DGAs with smallest number of states, starting from the canonical DFA. For DSAs we prove the following problem to be NP-complete in Section 8: given a DFA and a number k, does there exist a language equivalent DSA with total size at most k? This result exhibits an inherent difficulty in minimizing DSAs.

A method to derive small DSAs from a given DFA. Similar in spirit to the work on DGAs, we would like to be able to get small DSAs starting from a given DFA. Our main technical constribution is a procedure to derive language equivalent DSAs from a given complete DFA. Of course, the given complete DFA itself can be considered as a DSA. However, our goal is to generate DSAs with as small a total size as possible.

- The DFA-to-DSA derivation method is presented in Section 5. In a nutshell, the derivation procedure selects subsets of DFA-states, and adds transitions labeled with (some of) the acyclic paths between them. The technical challenge lies in identifying sufficient conditions on the selected subset of states, so that the derivation procedure preserves the language (Theorem 5.9).
- We remark that minimal DSAs need not be unique, and make a surprising observation: the smallest DSA that we derive from the canonical DFA of a language L need not be a minimal DSA. We find this surprising because (1) firstly, our derivation procedure is surjective: every DSA (satisfying some natural assumptions) can be derived from some corresponding DFA (Proposition 6.2), and in particular, a minimal DSA can be derived from some DFA; (2) the observation suggests that one may need to start with a bigger DFA in order to derive a minimal DSA – so, starting with a bigger DFA may result in a smaller DSA (Section 6).
- Inspired by the above observation, we present a restriction to the definition of DSA, called *strong DSA* (Section 7). We show that minimal strong DSAs can in fact be derived from the canonical DFA, through our derivation procedure. This provides more insights into our derivation procedure and in some sense explains better the capabilities of our procedure.

This introductory work on DSAs opens several threads for future research. We discuss a few of them in the Conclusion (Section 9). A shorter version of this work appeared in the conference proceedings [KSVV24]. In the current version, Section 7 is completely new and does not appear in the conference proceedings.

Related work. The closest to our work is [GM99] which introduces DGAs, and gives a procedure to derive DGAs from DFAs. The focus however is on getting DGAs with as few states as possible. The observations presented in Section 6 of our work, also apply for state-minimality: the same example shows that in order to get a DSA with fewer states, one may have to start with a bigger DFA. This is in sharp contrast to the DGA setting, where the derivation procedure of [GM99] yields a minimal DGA (in the number of states) when applied on the canonical DFA. The problem of deriving DGAs with minimal total-size was left open in [GM99], and continues to remain so, to the best of our knowledge. Expression automata [HW04] allow regular expressions as transition labels. This model was already considered in [BM63] to convert automata to regular expressions. Every DFA can be converted to a two state expression automaton with a regular expression connecting them. A model of deterministic Expression automata (DEA) was proposed in [HW04] with restrictions that limit the expressive power. An algorithm to convert a DFA to a DEA, by repeated state elimination, is proposed in [HW04]. The resulting DEA is minimal in the number of states. Minimization of NFAs was studied in [JR93] and shown to be hard. Succinctness of models with different features, like alternation, two-wayness, pebbles, and a notion of concurrency, has been studied in [GH96].

Here are some works that are related to the spirit of finding more readable specifications. [Fer09] has used the model of deterministic GA to develop a learning algorithm for some simple forms of regular expressions, with applications in learning DTD specifications for XML code. In this paper, the author talks about readability of specifications. They claim that regular expressions (REs) are arguably the best way to specify regular languages. They also claim that the translation algorithms from DFAs to REs give unreadable REs. In the paper, they consider specific language classes and generate algorithms to learn simple looking REs.

Another work that talks about readability of specifications is [ZLL02]. It considers automata based specification languages and performs extensive experiments to determine which choice of syntax gives better readability. The authors remark that hierarchical specifications are easier, since not every transition needs to be specified. Once again, we observe that there is an effort to remove transitions. Another formalism is proposed in [VSKA14] which uses a tabular notation where each cell contains patterns, which is then compiled into a usual automaton for analysis.

2. Preliminaries

We fix a finite alphabet Σ. Following standard convention, we write Σ^* for the set of all words (including ε) over Σ, and $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$. For $w \in \Sigma^*$, we write $|w|$ for the length of w, with $|\varepsilon|$ considered to be 0. A word u is a *prefix* of word w if $w = uv$ for some $v \in \Sigma^*$; it is a *proper-prefix* if $v \in \Sigma^+$. Observe that ε is a prefix of every word. A set of words W is said to be a *prefix-free set* if no word in W is a prefix of another word in W. A word u is a *suffix* (resp. *proper-suffix*) of w if $w = vu$ for some $v \in \Sigma^*$ (resp. $v \in \Sigma^+$). A word u is a *factor* of word w if it is a suffix of a proper-prefix of w, that is $w = v_1uv_2$ for some $v_1 \in \Sigma^*, v_2 \in \Sigma^+$.

A *Deterministic Finite Automaton (DFA)* M is a tuple $(Q, \Sigma, q^{init}, \delta, F)$ where Q is a finite set of states, $q^{init} \in Q$ is the initial state, $F \subseteq Q$ is a set of accepting states, and $\delta : Q \times \Sigma \to Q$ is a partial function describing the transitions. If δ is complete, the automaton is said to be a complete DFA. Else, it is called a trim DFA. The run of DFA M on a word $w = a_1a_2 \dots a_n$ (where $a_i \in \Sigma$) is a sequence of transitions $(q_0, a_1, q_1)(q_1, a_2, q_2) \dots (q_{n-1}, a_n, q_n)$

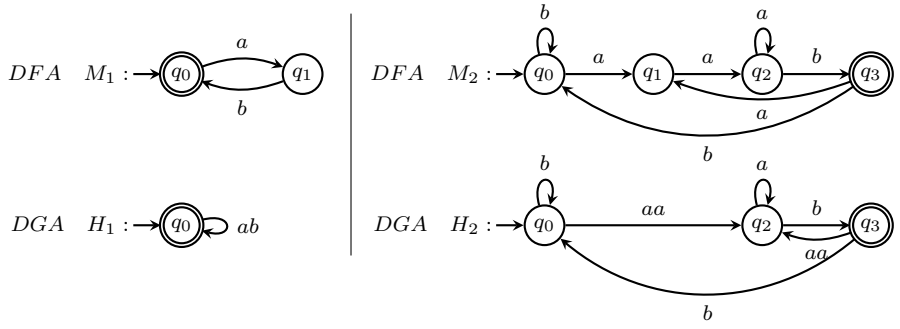

Figure 2: Examples of DFAs and corresponding DGAs, over alphabet $\{a, b\}$.

where $\delta(q_i, a_{i+1}) = q_{i+1}$ for each $0 \leq i < n$, and $q_0 = q^{init}$, the initial state of M. The run is accepting if $q_n \in F$. If the DFA is complete, every word has a unique run. On a trim DFA, each word either has a unique run, or it has no run. The language $\mathcal{L}(M)$ of DFA M, is the set of words for which M has an accepting run.

We will now recall some useful facts about minimality of DFAs. Here, by minimality, we mean DFAs with the least number of states. Every complete DFA M induces an equivalence $\sim_M$ over words: $u \sim_M v$ if M reaches the same state on reading both u and v from the initial state. In the case of trim DFAs, this equivalence can be restricted to set of prefixes of words in $\mathcal{L}(M)$. For a regular language L, we have the Nerode equivalence: $u \approx_L v$ if for all $w \in \Sigma^*$, we have $uw \in L$ iff $vw \in L$. By the well-known Myhill-Nerode theorem (see [HMU07] for more details), there is a canonical DFA M_L with the least number of states for L, and $\sim_{M_L}$ equals the Nerode equivalence $\approx_L$. Furthermore, every DFA M for L is a *refinement* of M_L: $u \sim_M v$ implies $u \sim_{M_L} v$. If two words reach the same state in M, they reach the same state in M_L.

A *Deterministic Generalized Automaton (DGA)* [GM99] H is given by $(Q, \Sigma, q^{init}, E, F)$ where Q, q^{init}, F mean the same as in DFA, and $E \subseteq Q \times \Sigma^+ \times Q$ is a finite set of edges labeled with words from Σ^+. For every state q, the set $\{\alpha \mid (q, \alpha, q') \in E\}$ is a prefix-free set. A run of DGA H on a word w is a sequence of edges $(q_0, \alpha_1, q_1)(q_1, \alpha_2, q_2) \dots (q_{n-1}, \alpha_n, q_n)$ such that $w = \alpha_1 \alpha_2 \dots \alpha_n$, with q_0 being the initial state. As usual, the run is accepting if $q_n \in F$. Due to the property of the set of outgoing labels being a prefix-free set, there is at most one run on every word. The language $\mathcal{L}(H)$ is the set of words with an accepting run. Figure 2 gives examples of DFAs and corresponding DGAs.

It was shown in [GM99] that there is no unique smallest DGA. The paper defines an operation to suppress states and create longer labels. A state of a DGA is called *superflous* if it is neither the initial nor final state, and it has no self-loop. For example, in Figure 2, in M_1 and M_2, state q_1 is superfluous. Such states can be removed, and every pair $p \xrightarrow{\alpha} q$ and $q \xrightarrow{\beta} r$ can be replaced with $p \xrightarrow{\alpha\beta} r$. This operation is extended to a set of states: given a DGA H, a set of states S, a DGA $\mathcal{S}(H, S)$ is obtained by suppressing states of S, one after the other, in any arbitrary order. For correctness, there should be no cycle in the induced subgraph of H restricted to S. The paper proves that minimal DGAs (in number of states) can be derived by suppressing states, starting from the canonical DFA.

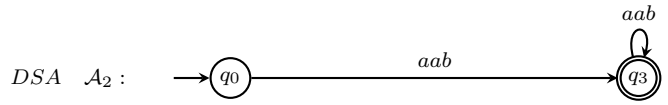

Figure 3: DSA $\mathcal{A}_2$ accepts $L_2 = \Sigma^* aab$, with $\Sigma = \{a, b\}$.

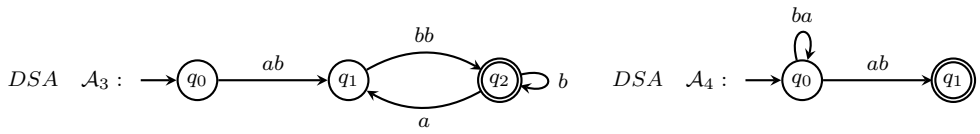

Figure 4: $\mathcal{A}_3$ accepts $L_3 = \Sigma^* ab \Sigma^* bb$ and $\mathcal{A}_4$ accepts $L_4 = (b^* ba)^* a^* ab$.

3. A new automaton model – DSA

We have seen an example of a deterministic suffix automaton in Figure 1. A DSA consists of a set of states, and a finite set of outgoing labels at each state. On an input word w, the DSA finds the earliest prefix which ends with an outgoing label of the initial state, erases this prefix and goes to the target state of the transition with the matching label. Now, the DSA processes the rest of the word from this new state in the same manner. In this section, we will formally describe the syntax and semantics of DSA.

We start with some more examples. Figure 3 shows a DSA for $L_2 = \Sigma^* aab$, the same language as the automata M_2 and H_2 of Figure 2. At q_0, DSA $\mathcal{A}_2$ waits for the first occurrence of aab and as soon as it sees one, it transitions to q_3. Here, it waits for further occurrences of aab. For instance, on the word $abbaabbbaab$, it starts from q_0 and reads until $abbaab$ to move to q_3. Then, it reads the remaining $bbaab$ to loop back to q_3 and accepts. On a word $baabaa$, the automaton moves to q_3 on $baab$, and continues reading aa, but having nowhere to move, it makes no transition and rejects the word. Consider another language $L_3 = \Sigma^* ab \Sigma^* bb$ on the same alphabet Σ. A similar machine (as $\mathcal{A}_2$) to accept L_3 would look like $\mathcal{A}_3$ depicted in Fig. 4. For example, on the word $abbbb$, it would read until ab and move from q_0 to q_1, read further until bb and move to q_2, then read b and move back to q_2 to accept. We can formally define such machines as automata that transition on suffixes, or suffix-reading automata.

Definition 3.1 (DSA)**.** A *deterministic suffix-reading automaton (DSA)* $\mathcal{A}$ is given by a tuple $(Q, \Sigma, q^{init}, \Delta, F)$ where Q is a finite set of states, Σ is a finite alphabet, $q^{init} \in Q$ is the initial state, $\Delta \subseteq Q \times \Sigma^+ \times Q$ is a finite set of transitions, $F \subseteq Q$ is a set of accepting states. For a state $q \in Q$, we define $\mathrm{Out}(q) := \{\alpha \mid (q, \alpha, q') \in \Delta \text{ for some } q' \in Q\}$ for the set of labels present in transitions out of q. No state has two outgoing transitions with the same label: if $(q, \alpha, q') \in \Delta$ and $(q, \alpha, q'') \in \Delta$, then $q' = q''$.

The (total) size $|\mathcal{A}|$ of DSA $\mathcal{A}$ is defined as the sum of the number of states, the number of transitions, and the size $|\,\mathrm{Out}(q)|$ for each $q \in Q$, where $|\,\mathrm{Out}(q)| := \sum_{\alpha \in \mathrm{Out}(q)} |\alpha|$.

As mentioned earlier, at a state q the automaton waits for a word that ends with one of its outgoing labels. If more than one label matches, then the transition with the longest label is taken. For example, consider the DSA in Figure 1. At state s_1 on reading $fghendif$, both the `if` and `endif` transitions match. The longest match is `endif` and therefore the DSA moves to s_0. This gives a deterministic behaviour to the DSA. More precisely: at a state q, it reads w to fire (q, α, q') if α is the longest word in $\mathrm{Out}(q)$ which is a suffix of w,

and no proper prefix of w has any label in $\mathrm{Out}(q)$ as suffix. We call this a 'move' of the DSA. For example, consider $\mathcal{A}_4$ of Figure 4 as a DSA. Let us denote $t := (q_0, ab, q_1)$ and $t' := (q_0, ba, q_1)$. We have moves (t, ab), (t, aab), $(t, aaab)$, and (t', ba), (t', bba), etc. In order to make a move on t, the word should end with ab and should have neither ab nor ba in any of its proper prefixes.

Definition 3.2. A *move* of DSA $\mathcal{A}$ is a pair (t, w) where $t = (q, \alpha, q') \in \Delta$ is a transition of $\mathcal{A}$ and $w \in \Sigma^+$ such that

- α is the longest word in $\mathrm{Out}(q)$ which is a suffix of w, and
- no proper prefix of w has a label in $\mathrm{Out}(q)$ as suffix (no label is a factor of w).

A move (t, w) denotes that at state q, transition t gets triggered on reading word w. We will also write $q \xrightarrow[\alpha]{w} q'$ for the move (t, w).

Whether a word is accepted or rejected is determined by a 'run' of the DSA on it.

Definition 3.3. A run of $\mathcal{A}$ on word w, starting from a state q, is a sequence of moves that consume the word w, until a (possibly empty) suffix of w remains for which there is no move possible:

formally, a run is a sequence of moves $q_i \xrightarrow[\alpha_i]{w_i} q_{i+1}$ for $0 \leq i \leq m$ and $q_m \xrightarrow{w_m}$ where $q = q_0$ and m is the index of the last state in the sequence i.e. $q = q_0 \xrightarrow[\alpha_0]{w_0} q_1 \xrightarrow[\alpha_1]{w_1} \cdots \xrightarrow[\alpha_{m-1}]{w_{m-1}} q_m \xrightarrow{w_m}$ such that $w = w_0 w_1 \dots w_{m-1} w_m$, and $q_m \xrightarrow{w_m}$ denotes that there is no move using any outgoing transition from q_m on w_m or any of its prefixes.

The run is accepting if $q_m \in F$ and $w_m = \varepsilon$ (no dangling letters in the end). The language $\mathcal{L}(\mathcal{A})$ of $\mathcal{A}$ is the set of all words that have an accepting run starting from the initial state q^{init}.

Naturally, the set of words with accepting runs gives the language of the DSA. Moreover, due to our "move" semantics, there is a unique run for every word.

4. Comparison with DFA and DGA

Every complete DFA can be seen as an equivalent DSA — since $\mathrm{Out}(q) = \Sigma$ for every state, the equivalent DSA is forced to move on each letter, behaving like the DFA that we started off with. For the DSA-to-DFA direction, we associate a specific DFA to every DSA, as follows. The idea is to replace transitions of a DSA with a string-matching-DFA for $\mathrm{Out}(q)$ at each state. Figure 5 gives an example. The intermediate states correspond to proper prefixes of words in $\mathrm{Out}(q)$.

Definition 4.1 (Tracking DFA for a DSA.)**.** For a DSA $\mathcal{A} = (Q^{\mathcal{A}}, \Sigma, q_{in}^{\mathcal{A}}, \Delta^{\mathcal{A}}, F^{\mathcal{A}})$, we give a DFA $M_{\mathcal{A}}$, called its *tracking DFA*. For $q \in Q^{\mathcal{A}}$, let $\overline{\mathrm{Out}}(q)$ be the set of all prefixes of words in $\mathrm{Out}(q)$. States of $M_{\mathcal{A}}$ are given by: $Q^M = \bigcup_{q \in Q^{\mathcal{A}}} \{\{(q, \beta) \mid \beta \in \overline{\mathrm{Out}}(q)\} \cup \{(q, \overline{\varepsilon})\}\}$, where $\overline{\varepsilon}$ is used as a special symbol to create a copy of each (q, ε).

The initial state is $(q_{in}^{\mathcal{A}}, \varepsilon)$ and final states are $\{(q, \varepsilon) \mid q \in F^{\mathcal{A}}\}$. Transitions are as below: for every $q \in Q^{\mathcal{A}}, \beta \in \overline{\mathrm{Out}}(q), a \in \Sigma$, let β' be the longest word in $\overline{\mathrm{Out}}(q)$ s.t β' is a suffix of βa.

- $(q, \beta) \xrightarrow{a} (q', \varepsilon)$ if $\beta' \in \mathrm{Out}(q)$ and $(q, \beta', q') \in \Delta^{\mathcal{A}}$,
- $(q, \beta) \xrightarrow{a} (q, \beta')$ if $\beta' \notin \mathrm{Out}(q)$ and $\beta' \neq \varepsilon$,

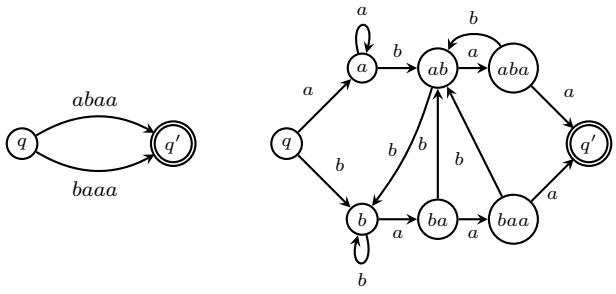

Figure 5: A DSA on the left, and the corresponding DFA for matching the strings *abaa* and *baaa*.

- $(q, \beta) \xrightarrow{a} (q, \overline{\varepsilon})$ if $\beta' = \varepsilon$,
- $(q, \overline{\varepsilon}) \xrightarrow{a} s$, if $(q, \varepsilon) \xrightarrow{a} s$ according to the above (same outgoing transitions).

Intuitively, the tracking DFA implements the transition semantics of DSAs. Starting at (q, ε), the tracking DFA moves along states marked with q as long as no label of $\mathrm{Out}(q)$ is seen as a suffix. For all such words, the tracking DFA maintains the longest word among $\overline{\mathrm{Out}}(q)$ seen as a suffix so far. For instance, in Figure 5, at q on reading word *aab*, the DFA on the right is in state *ab* (which is the equivalent of (q, ab) in the tracking DFA definition).

In the rest of the document, we will make use of the following notation:

Definition 4.2 (Notation)**.**

- For two words w_1, w_2, we write $w_1 \sqsubseteq_{\mathsf{sf}} w_2$ if w_1 is a suffix of w_2.
- For a set of words $U \subseteq \Sigma^*$ and $\alpha, \beta \in \Sigma^*$, $\alpha \sqsubseteq^{U}_{\mathsf{lsf}} \beta$ if α is the longest word in U which is a suffix of β, i.e. for all $\alpha' \in U$ if $\alpha' \sqsubseteq_{\mathsf{sf}} \beta$ then $|\alpha'| \leq |\alpha|$.

Lemma 4.3. *For every DSA $\mathcal{A}$, the language $\mathcal{L}(\mathcal{A})$ equals the language $\mathcal{L}(M_{\mathcal{A}})$ of its tracking DFA.*

Proof. The tracking DFA satisfies the following invariants on every state (q, ε): (we make use of notations from Definition 4.2)

- Let w be a word that has no $\alpha \in \mathrm{Out}(q)$ as its factor. Then, the run of w starting at (q, ε), remains in states of the form (q, β) and ends in state (q, β_w) where $\beta_w \sqsubseteq^{\overline{\mathrm{Out}}(q)}_{\mathsf{lsf}} w$, when $\beta_w \neq \varepsilon$; if $\beta_w = \varepsilon$, then it ends in state $(q, \overline{\varepsilon})$.
- Let w be a word such that $q \xrightarrow[\alpha]{w} q'$ is a move of $\mathcal{A}$. Then the run of $M_{\mathcal{A}}$ on w starting at (q, ε) remains in states of the form (q, β) and ends in (q', ε).
- Let w be a word such that the run of $M_{\mathcal{A}}$ starting at (q, ε) remains in states of the form (q, β) and ends in a state (q, β_w). If $\beta_w \neq \overline{\varepsilon}$, then $\beta_w \sqsubseteq^{\overline{\mathrm{Out}}(q)}_{\mathsf{lsf}} w$. If $\beta_w = \varepsilon$, then there is no suffix of w in $\overline{\mathrm{Out}}(q)$.
- Let w be a word such that the run of $M_{\mathcal{A}}$ starting at (q, ε) ends in (q', ε), and all intermediate states are of the form (q, β). Then there is a move $q \xrightarrow[\alpha]{w} q'$ in $\mathcal{A}$ where $\alpha \sqsubseteq^{\mathrm{Out}(q)}_{\mathsf{lsf}} w$.

All these invariants can be proved by induction on the length of the word w and making use of the way the transitions have been defined in the tracking DFA. The first two invariants show that every accepting run of $\mathcal{A}$ has a corresponding accepting run in $M_{\mathcal{A}}$, thereby proving

$\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(M_{\mathcal{A}})$. The next two invariants show that every accepting run of $M_{\mathcal{A}}$ corresponds to an accepting run of $\mathcal{A}$, thereby showing $\mathcal{L}(M_{\mathcal{A}}) \subseteq \mathcal{L}(\mathcal{A})$. □

Lemma 4.3 and the fact that every complete DFA is also a DSA, prove that DSAs recognize regular languages. We will now compare succinctness of DSA wrt DFA and DGA, starting with a family of languages for which DSAs are concise.

Lemma 4.4. *Let $\Sigma = \{a_1, a_2, \ldots, a_n\}$ for some $n \geq 1$. Consider the language $L_n = \Sigma^* a_1 a_2 \ldots a_n$. There is a DSA for this language with size $(4 + 2|\Sigma|)$. Any trim DFA for L_n has size at least $|\Sigma|^2$. (Note: size is counted as the sum of the number of states, edges and length of edge labels, in all the automata)*

Proof. Consider the DSA with states q_0, q_1 and transitions $q_0 \xrightarrow{a_1 \ldots a_n} q_1$, $q_1 \xrightarrow{a_1 \ldots a_n} q_1$, with q_0 the initial state and q_1 the accepting state. This DSA accepts L_n.

For any pair $a_1 a_2 \ldots a_i$ and $a_1 a_2 \ldots a_j$ with $i \neq j$, there is a distinguishing suffix: $a_1 a_2 \ldots a_i \cdot a_{i+1} \ldots a_n \in L_n$, but $a_1 a_2 \ldots a_j \cdot a_{i+1} \ldots a_n \notin L_n$. Therefore, the strings a_1, $a_1 a_2$, $\ldots$, $a_1 \ldots a_n$ go to different states in the canonical DFA. This shows there are at least n states in the minimal DFA. Now, from $a_1 a_2 \ldots a_j$, there is a transition of every letter: clearly, there needs to be a transition on a_{j+1}; if there is no transition on, say $a_\ell \neq a_{j+1}$, then the word $a_1 a_2 \ldots a_j a_\ell a_1 a_2 \ldots a_n$ will be rejected. A contradiction. Hence, there are n transitions from each state. □

We make a remark about DGAs for the language family in Lemma 4.4. Notice that suppressing superfluous states of the minimal DFA leads to a strict increase in the label length, which only strictly increases the (total) size. For instance, suppose $q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} q_3$ is a sequence of transitions in the minimal DFA for L_n. There are additional transitions $q_2 \xrightarrow{a_1} q_1$ and $q_2 \xrightarrow{\Sigma \setminus \{a_1, a_3\}} q_0$ (where q_0 is the initial state). Suppressing q_2 leads to edges with labels $a_2 a_3$, $a_2 a_1$, $a_2 c$ for each $c \in \Sigma \setminus \{a_1, a_3\}$. Notice that a_2 gets copied $|\Sigma|$ many times. Therefore suppressing states from the minimal DFA is not helpful in getting a smaller DGA. However, using this argument, we cannot conclude that the minimal DGA has size at least n^2. Since a characterization of minimality of DGAs in terms of total size is not known, we leave it at this remark.

We now state the final result of this section, which summarizes the size comparison between DSAs, DFAs, DGAs. For the comparison to DFAs, we use the fact that every DSA of size k can be converted to its tracking DFA, which has atmost $2k$ states. Therefore, size of the tracking DFA is bounded by $2k$ (states) $+2k \cdot |\Sigma|$ (edges) $+2k \cdot |\Sigma|$ (label length), which comes to $2k(1 + 2|\Sigma|)$.

For a regular language L, let $n_F^{cmp}(L), n_F^{trim}(L), n_G^{trim}(L), n_S(L)$ denote the size of the minimal complete DFA, minimal trim DFA, minimal trim DGA and minimal DSA respectively, where size is counted as the sum of the number of states, edges and length of edge labels, in all the automata.

Theorem 4.5. *We have:*

(1) *For all L, we have* $\dfrac{n_F^{cmp}(L)}{2(1 + 2|\Sigma|)} \leq n_S(L) \leq n_F^{cmp}(L)$

(2) *There is a language for which $n_S(L)$ is the smallest, and another language for which $n_S(L)$ is the largest of the three.*

Proof. A complete DFA can be seen as a DSA accepting the same language. This gives us $n_S(L) \leq n_F^{cmp}(L)$. For the inequality $\frac{n_F^{cmp}(L)}{2(1+2|\Sigma|)} \leq n_S(L)$, we show that the tracking DFA (Definition 4.1) of a DSA has size at most $n_S(L) \cdot 2(1+2|\Sigma|)$. The number of states of the tracking DFA is atmost $2n_S(L)$ (recall that $n_S(L)$ denotes the total size of the DSA, which includes the label lengths). For each state there are Σ transitions. Therefore, total size is $2n_S(L)$ (states) + $2n_S(L) \cdot |\Sigma|$ (edges) + $2n_S(L) \cdot |\Sigma|$ (label lengths), which equals $2n_S(L)(1+2|\Sigma|)$.

For the second part of the proof, Lemma 4.4 gives an example where $n_S(L)$ is the smallest. For the other direction, consider the language $L_1 = (ab)^*$. A trim DFA for this language has two states q_0, q_1 (with q_0 accepting), and two transitions $q_0 \xrightarrow{a} q_1$, and $q_1 \xrightarrow{b} q_0$. Therefore $n_F^{trim}(L_1) \leq 2+2+2 = 6$. A trim DGA for this language is $q_0 \xrightarrow{ab} q_0$. Therefore $n_G^{trim}(L_1) \leq 2+1+2 = 5$. We first claim that any DSA for this language needs to maintain the following information: (1) the initial state which is accepting, as ε is in the language; (2) there is a path from the initial state on ab to an accepting state (otherwise ab will not be accepted); (3) on reading aa or b from the initial state, the DSA has to make some transitions and go to a sink state, which is a non-accepting state — otherwise, the words aab or bab will get accepted. This shows that any DSA has at least two states, at least three transitions, and ab, b and aa split across some transition labels. This gives either: $n_S(L_1) \geq 2+2+4 = 8$ for the DSA with two states and $\xrightarrow{ab}$, $\xrightarrow{b}$ and $\xrightarrow{aa}$ as transitions, which is bigger than the corresponding trim DFA and DGA, or the min DFA viewed as DSA, with 3 states, and transitions on each of a and b from each of the states giving $n_S \geq 3+6+6 = 15$ □

5. Suffix-tracking sets – obtaining DSA from DFA

In this section, we are interested in the following task: given a DFA, extract a language equivalent DSA with as small a total size as possible. For instance, given the DFA on the right of Figure 5, how do we get the DSA in the left of the same figure? In some sense, our method would be to reverse engineer the tracking DFA construction that was introduced in Section 4, culminating in Definition 4.1.

For DGAs, a method to derive smaller DGAs by suppressing states was recalled in Section 2. The DSA model creates new challenges. Suppressing states may not always lead to smaller automata (in total size). Figure 6 illustrates an example where suppressing states leads to an exponentially larger automaton, due to the exponentially many paths created. But, suppressing states may sometimes indeed be useful: in Figure 7, the DFA on the left is performing a string matching to deduce the pattern ab. On seeing ab, it accepts. Any extension is rejected. This is succinctly captured by the DSA on the right. Notice that the DSA is obtained by suppressing states q_1 and q_3. So, suppressing states may sometimes be useful and sometimes not. In [GM99], the focus was on getting a DGA with minimal number of states, and hence suppressing states was always useful.

More importantly, when can we suppress states? DGAs cannot "ignore" parts of the word. This in particular leads to the requirement that a state with a self-loop cannot be suppressed. DSAs have a more sophisticated transition semantics. Therefore, the procedure to suppress states is not as simple. This is the subject of this section. We deviate from the DGA setting in two ways: we will select a subset of good states from which we can construct a DSA (essentially, this means the rest of the states are suppressed); secondly, our starting

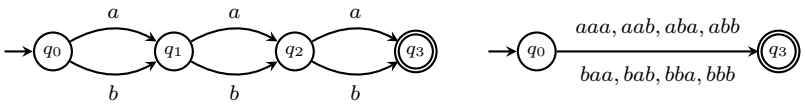

Figure 6: Suppressing states can add exponentially many labels and increase total size.

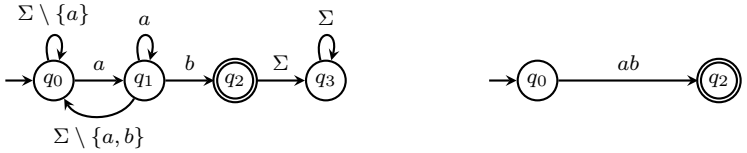

Figure 7: Suppressing states can sometimes reduce total size

point will be complete DFA, on which we make the choice of states (in DGAs, one could start with any DGA and suppress states). Our procedure can be broken down into two steps: (1) Start from a complete DFA, select a subset of states and build an induced DSA by connecting states using acyclic paths between them; (2) Remove some redundant transitions.

The plan for this section is as follows.

Section 5.1: We present the core technical concept of suffix-tracking sets (Definition 5): when a set S of states of a given complete DFA M is suffix-tracking, a language equivalent DSA can be induced from M. For instance, in Figure 5, the set $\{q, q'\}$ would be suffix-tracking for the DFA on the right. The DSA induced from $\{q, q'\}$ would contain simple paths from q to q' in M.

Section 5.2: The induced DSA from suffix-tracking sets can contain useless transitions. For instance, in the same Figure 5, in addition to simple paths *abaa* and *baaa*, there are paths *abbaaa*, *babaa* that move across different branches in the DFA. These are useless transitions, since there are smaller suffixes *baaa* and *abaa* respectively that match these patterns. One cannot always simply remove them. Some care needs to be taken while removing transitions. We discuss these observations in this section.

5.1. **Building an induced DSA.** We start with an illustrative example. Consider DFA M in Figure 8. The DSA on the right of the figure shows such an induced DSA obtained by marking states $\{q_0, q_2\}$ and connecting them using simple paths. Notice that the language of the induced DSA and the original DFA are same in this case. Intuitively, all words that end with an a land in q_1. Hence, q_1 can be seen to "track" the suffix a. Now, consider Figure 9. We do the same trick, by marking states $\{q_0, q_2\}$ and inducing a DSA. Observe that the DSA does not accept *aba*, and hence is not language equivalent. When does a subset of states induce a language equivalent DSA? Roughly, this is true when the states that are suppressed track "suitable suffixes" (a reverse engineering of the tracking DFA construction of Definition 4.1). As we will see, the suitable suffixes will be the simple paths from the selected states to the suppressed states. We begin by formalizing these ideas and then present sufficient conditions that ensure language equivalence of the resulting DSA.

Definition 5.1 (Simple words)**.** Consider a complete DFA $M = (Q, \Sigma, q^{init}, \Delta, F)$. Let $S \subseteq Q$ be a subset of states, and $p, q \in Q$. We define $\mathsf{SP}(p \rightsquigarrow q, S)$, the *simple words*

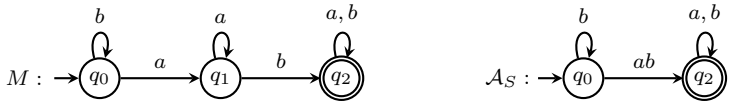

Figure 8: DFA M and an equivalent DSA $\mathcal{A}_S$ 'induced' with $S = \{q_0, q_2\}$.

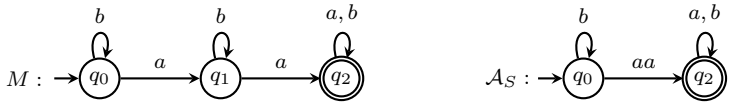

Figure 9: DFA M and DSA $\mathcal{A}_S$ 'induced' with $S = \{q_0, q_2\}$. Not equivalent.

from p to q modulo S, as the set of all words $a_1 a_2 \dots a_n \in \Sigma^+$ such that there is a path: $p = p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} \cdots p_{n-1} \xrightarrow{a_n} p_n = q$ in M where

- no intermediate state belongs to S: $\{p_1, \dots, p_{n-1}\} \subseteq Q \setminus S$, and
- there is no intermediate cycle: if $p_i = p_j$ for some $0 \leq i < j \leq n$, then $p_i = p_0$ and $p_j = p_n$.

We write $\mathsf{SP}(p, S)$ for $\bigcup_{q \in Q} \mathsf{SP}(p \rightsquigarrow q, S)$, the set of all simple words modulo S, emanating from p.

For example, in Figure 8, with $S = \{q_0, q_2\}$, we have $\mathsf{SP}(q_0 \rightsquigarrow q_1, S) = \{a\}$, $\mathsf{SP}(q_0 \rightsquigarrow q_0, S) = \{b\}$ and $\mathsf{SP}(q_0 \rightsquigarrow q_2, S) = \{ab\}$. These are the same in Figure 9, except $\mathsf{SP}(q_0 \rightsquigarrow q_2, S) = aa$.

Here are some preliminary technical lemmas. Due to determinism of M, we get the following property, which underlies several arguments that come later.

Lemma 5.2. *Let $S \subseteq Q$, and $p, q, r \in Q$ s.t. $q \neq r$. We have $\mathsf{SP}(p \rightsquigarrow q, S) \cap \mathsf{SP}(p \rightsquigarrow r, S) = \emptyset$.*

The next lemma says that every transition either extends a simple path or is a "back-edge" leading to an ancestor in the path.

Lemma 5.3. *Let $S \subseteq Q$ and $p, q, u \in Q$ ($q \notin S, p \neq q$) such that $q \xrightarrow{a} u$ is a transition. For every $\sigma \in \mathsf{SP}(p \rightsquigarrow q, S)$, either σa belongs to $\mathsf{SP}(p \rightsquigarrow u, S)$, or some proper prefix of σa belongs to $\mathsf{SP}(p \rightsquigarrow u, S)$.*

Proof. Since $\sigma \in \mathsf{SP}(p \rightsquigarrow q, S)$, there is a path $p = p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} \cdots p_{n-1} \xrightarrow{a_n} p_n = q$, with $\sigma = a_1 a_2 \dots a_n$, satisfying the conditions of Definition 5.1. If $u \neq p_i$ for $0 \leq i \leq n$, then $\sigma a \in \mathsf{SP}(p \rightsquigarrow u, S)$ since $q \notin S$. If $u = p_i$ for some $0 < i \leq n$, then the prefix $a_1 a_2 \dots a_{i-1} \in \mathsf{SP}(p \rightsquigarrow u, S)$. If $u = p_0$, then we have $\sigma a \in \mathsf{SP}(p \rightsquigarrow u, S)$. □

Fix a complete DFA M for this section. A DSA can be 'induced' from M using S, by fixing states to be S (initial and final states retained) and transitions to be the simple words modulo S connecting them i.e. $p \xrightarrow{\sigma} q$ if $\sigma \in \mathsf{SP}(p \rightsquigarrow q, S)$ (Figure 8).

Definition 5.4 (Induced DSA)**.** Given a DFA M and a set S of states in M that contains the initial and final states, we define the induced DSA of M (using S). The states of the induced DSA are given by S. The initial and final states are the same as in M. The transitions are given by the simple words modulo S i.e. $p \xrightarrow{\sigma} q$ if $\sigma \in \mathsf{SP}(p \rightsquigarrow q, S)$, for every pair of states $p, q \in S$.

The induced DSA may not be language-equivalent (Figure 9); to ensure that, we need to check some conditions. Here is a central definition.

Definition 5.5 (Suffix-compatible transitions)**.** Fix a subset $S \subseteq Q$. A transition $q \xrightarrow{a} u$ is suffix-compatible w.r.t. S if either $q \in S$ or $u \in S$ **OR** for all $p \in S$, and for every $\sigma \in \mathsf{SP}(p \rightsquigarrow q, S)$, there is an $\alpha \in \mathsf{SP}(p \rightsquigarrow u, S)$ s.t.:

- α is a suffix of σa, and
- moreover, α is the longest suffix of σa among words in $\mathsf{SP}(p, S)$.

Note that a transition $q \xrightarrow{a} u$ is trivially suffix-compatible if $q \in S$ or $u \in S$. The rest of the condition only needs to be checked when both of $q, u \notin S$. In Figure 9, we find the self-loop at q_1 to not be suffix-compatible: we have $S = \{q_0, q_2\}$, and $\mathsf{SP}(q_0 \rightsquigarrow q_1, S) = \{a\}$, $\mathsf{SP}(q_0, S) = \{b, a, ab\}$; the transition $q_1 \xrightarrow{b} q_1$ is not suffix-compatible since there is no suffix of ab in $\mathsf{SP}(q_0 \rightsquigarrow q_1, S)$. Whereas in Figure 8, the loop is labeled a instead of b. The transition $q_1 \xrightarrow{a} q_1$ is suffix-compatible, since the longest suffix of aa among $\mathsf{SP}(q_0, S)$ is a and it is present in $\mathsf{SP}(q_0 \rightsquigarrow q_1, S)$. Let us take the DFA in the right of Figure 5, and let $S = \{q, q'\}$. Here are some of the simple path sets: $\mathsf{SP}(q \rightsquigarrow ab, S) = \{ab, bab, baab\}$, $\mathsf{SP}(q \rightsquigarrow aba, S) = \{aba, baba, baaba\}$. Consider the transition $aba \xrightarrow{b} ab$. It can be verified that for every $\sigma \in \mathsf{SP}(q \rightsquigarrow aba, S)$, the longest suffix of the extension σb, among simple paths out of q, indeed lies in the state ab. In fact, all transitions satisfy suffix-compatibility w.r.t. the chosen set S.

The suffix-compatibility condition is described using simple paths to states. It requires that every transition take each simple word reaching its source to the state tracking the longest suffix of its one-letter extension. This condition on simple paths, transfers to all words, that circle around the suppressed states. In Figure 5, this property can be verified by considering the word $bbabab$ and its run: $q \xrightarrow{b} b \xrightarrow{b} b \xrightarrow{a} ba \xrightarrow{b} ab \xrightarrow{a} aba \xrightarrow{b} ab$. At each step, the state reached corresponds to the longest suffix among the simple words out of q. In the next two lemmas, we prove this claim.

We will use a special notation: for a state $p \in S$, we write $\mathrm{Out}(p, S)$ for $\bigcup_{r \in S} \mathsf{SP}(p \rightsquigarrow r, S)$; these are the simple words that start at p and end in some state r of S. Notice that these are the words that appear as transitions in the induced DSA. In particular, $\mathrm{Out}(p)$ in the induced DSA equals $\mathrm{Out}(p, S)$. We also remark that $\mathrm{Out}(p, S)$ is different from $\mathsf{SP}(p, S)$: the latter considers simple words from p to all states in Q, whereas the former only considers words from p to S.

Lemma 5.6. *Let S be a set of states such that every transition of M is suffix-compatible w.r.t. S. Pick $p \in S$, and let $w \in \Sigma^+$ be a word with a run $p = p_0 \xrightarrow{w_1} p_1 \xrightarrow{w_2} p_2 \dots p_{n-1} \xrightarrow{w_n} p_n$ such that the intermediate states $p_1, \dots, p_{n-1}$ belong to $Q \setminus S$. The state p_n may or may not be in S. Then:*

- *no proper prefix of w has any word from $\mathrm{Out}(p, S)$ as suffix (no factor of w is in $\mathrm{Out}(p, S)$), and*
- *there is $\alpha \in \mathsf{SP}(p \rightsquigarrow p_n, S)$ such that α is the longest suffix of w among words in $\mathsf{SP}(p, S)$.*

Proof. We prove this by induction on the length of the word w. When $w = a$ for $a \in \Sigma$, we have the run $p \xrightarrow{a} q$. The first conclusion of the lemma is vacuously true, since there is no non-empty proper prefix of a. For the second conclusion, note that, by definition, we have $a \in \mathsf{SP}(p \rightsquigarrow q, S)$ (in both cases when $q \neq p$ and $q = p$). Moreover, clearly a is the longest suffix of a.

Now, let $w = w'a$ with a run $p \xrightarrow{w'} p' \xrightarrow{a} p_n$ such that p' and the intermediate states while reading w' belong to $Q \setminus S$. Assume the lemma holds for w'. Therefore, the longest suffix σ' of w', among $\mathsf{SP}(p, S)$, belongs to $\mathsf{SP}(p \rightsquigarrow p', S)$, that is: $\sigma' \sqsubseteq_{\mathsf{lsf}}^{\mathsf{SP}(p,S)} w'$ and $\sigma' \in \mathsf{SP}(p \rightsquigarrow p', S)$ (notation as in Definition 4.2). We claim that the longest suffix of $w'a$ is in fact the longest suffix of $\sigma'a$. If not: there is $\sigma''a$ with $|\sigma''| > |\sigma'|$ such that $\sigma''a \sqsubseteq_{\mathsf{lsf}}^{\mathsf{SP}(p,S)} w'a$. Therefore $\sigma'' \sqsubseteq_{\mathsf{lsf}}^{\mathsf{SP}(p,S)} w'$. This contradicts $\sigma' \sqsubseteq_{\mathsf{lsf}}^{\mathsf{SP}(p,S)} w'$. If $p_n \in Q \setminus S$, the longest suffix of $\sigma'a$ lies in p_n, by suffix-compatibility (Definition 5.5). If $p_n \in S$, we will have the exact word $\sigma'a \in \mathsf{SP}(p \rightsquigarrow p_n, S)$. □

Lemma 5.7. *Let S be a set of states such that every transition of M is suffix-compatible w.r.t. S. Let $p \in S$, and $w \in \Sigma^+$ be a word such that no proper prefix of w has a word from* $\mathrm{Out}(p, S)$ *as suffix (no factor of w is in* $\mathrm{Out}(p, S)$*). Then:*

- *The run of M starting from p, is of the form $p \xrightarrow{w_1} p_1 \xrightarrow{w_2} p_2 \dots p_{n-1} \xrightarrow{w_n} p_n$ where $\{p_1, \dots, p_{n-1}\} \subseteq Q \setminus S$ (notice that we have not included p_n, which may or may not be in S).*
- *the longest suffix of w, among $\mathsf{SP}(p, S)$ lies in $\mathsf{SP}(p \rightsquigarrow p_n, S)$.*

Proof. We prove this by induction on the length of the word w. Suppose $w = a$, for $a \in \Sigma$. Since M is complete, there is a transition $p \xrightarrow{a} p_1$. The first item is vacuously true. Moreover, if there is $q \in S$ and $\alpha \in \mathsf{SP}(p \rightsquigarrow q, S)$ such that α is a suffix of a, then clearly $\alpha = a$. Hence $q = p_1$, due to the determinism of the underlying automaton. If there is no such q, then it means $p_1 \notin S$.

Suppose $w = w'a$ satisfies the assumptions of the lemma. Assuming the lemma holds for w', we have a run $p \xrightarrow{w'} p'$ such that p' and all intermediate states lie in $Q \setminus S$. Let $p' \xrightarrow{a} p_n$ be the outgoing transition on a from p'. This gives a path $p \xrightarrow{w'} p \xrightarrow{a} p_n$ satisfying the first part of the lemma. By the second item of Lemma 5.6, the longest suffix $\alpha'a$ of $w'a$ among $\mathsf{SP}(p, S)$ belongs to $\mathsf{SP}(p \rightsquigarrow p_n, S)$. □

Suffix-compatibility alone does not suffice to preserve the language. In Figure 10, consider $S = \{0, 2, 4\}$. Every transition is suffix-compatible w.r.t. S. The DSA induced using S is shown in the middle. Notice that it is not language equivalent, due to the word aba for instance. The run of aba looks as follows: $0 \xrightarrow{ab} 4 \xrightarrow{b} 4$. The expected run was $0 \xrightarrow{aba} 2$, but that does not happen since there is a shorter prefix with a matching transition. Even though, we have suffix-compatibility, we need to ensure that there are no "conflicts" between outgoing patterns. This leads to the next definition.

Definition 5.8 (Well-formed set)**.** A set of states $S \subseteq Q$ is well-formed if there is no $p \in S, q \in S$ and $q' \notin S$, with a pair of words $\alpha \in \mathsf{SP}(p \rightsquigarrow q, S)$ (simple word to a state in S) and $\beta \in \mathsf{SP}(p \rightsquigarrow q', S)$ (simple word to a state not in S) such that α is a suffix of β.

We observe that the set $S = \{0, 2, 4\}$ is not well-formed since $b \in \mathsf{SP}(0 \rightsquigarrow 4, S), ab \in \mathsf{SP}(0 \rightsquigarrow 3, S)$ and b is a suffix of ab. Whereas $S' = \{0, 2, 3, 4\}$ is both suffix-tracking, and well-formed, and induces an equivalent DSA. On the word aba, the run on the DSA would be $0 \xrightarrow{ab} 3 \xrightarrow{a} 2$. The first move $0 \xrightarrow{ab} 3$ applies the longest match criterion, and fires the ab transition since ab is a longer suffix than b. This was not possible before since $3 \notin S$. It turns out that the two conditions — suffix-compatibility and well-formedness — are sufficient to induce a language equivalent DSA.

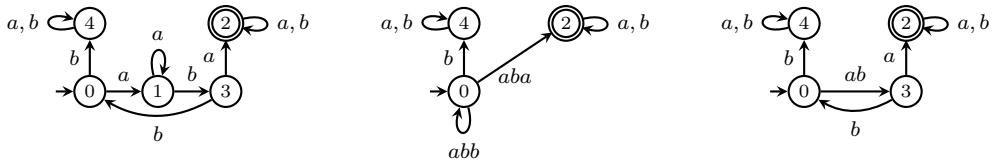

Figure 10: A DFA, a non-equivalent DSA and an equivalent induced DSA.

Definition 5.9 (Suffix-tracking sets)**.** A set of states $S \subseteq Q$ is suffix-tracking if it contains the initial and accepting states, and

(1) every transition of M is suffix-compatible w.r.t. S,
(2) and S is well-formed.

All these notions lead to the main theorem of this section.

Theorem 5.10. *Let S be a suffix-tracking set of complete DFA M, and let $\mathcal{A}_S$ be the DSA induced using S. Then: $\mathcal{L}(\mathcal{A}_S) = \mathcal{L}(M)$*

Proof. Pick $w \in \mathcal{L}(M)$. There is an accepting run $q_0 \xrightarrow{w_1} q_1 \xrightarrow{w_2} \dots \xrightarrow{w_n} q_n$ of M on w. By Definition 5.9, we have $q_0, q_n \in S$. Let $1 \leq i \leq n$ be the smallest index greater than 0, such that $q_i \in S$. Consider the run segment $q_0 \xrightarrow{w_1} q_1 \xrightarrow{w_2} \dots \xrightarrow{w_i} q_i$. By Lemma 5.6, and by the definition of induced DSA 5.4, no transition of $\mathcal{A}_S$ out of q_0 is triggered until $w_1 \dots w_{i-1}$, and then on reading w_i, the transition $q_0 \xrightarrow{\alpha} q_i$ is triggered, where $\alpha \in \mathsf{SP}(p \rightsquigarrow q, S)$, and α is also the longest suffix of $w_1 \dots w_i$ among $\mathsf{SP}(p, S)$. In particular, it is the longest suffix among outgoing labels from q_0 in $\mathcal{A}_S$. This shows there is a move $q_0 \xrightarrow[\alpha]{w_1 \dots w_i} q_i$ in $\mathcal{A}_S$. Repeat this argument on rest of the run $q_i \xrightarrow{w_{i+1}} q_{i+1} \xrightarrow{w_{i+1}} \dots \xrightarrow{w_n} q_n$ to extend the run of $\mathcal{A}_S$ on the rest of the word. This shows $w \in \mathcal{L}(\mathcal{A}_S)$.

Pick $w \in \mathcal{L}(\mathcal{A}_S)$. There is an accepting run ρ of $\mathcal{A}_S$ starting at the initial state q_0. Consider the first move $q_0 \xrightarrow[\alpha]{w_1 \dots w_i} q_i$ of $\mathcal{A}_S$ on the word. By the semantics of a move (Definition 3.2) and Lemma 5.7, we obtain a run $q_0 \xrightarrow{w_1} q_1 \xrightarrow{w_2} \dots q_{i-1} \xrightarrow{w_i} q_i$ of M where the intermediate states $q_1, \dots, q_{i-1}$ lie in $Q \setminus S$. We apply this argument for each move ρ in the accepting run of $\mathcal{A}_S$ to get an accepting run of M. □

We extend the idea of well-formed set of states to a corresponding notion on DSAs. We will employ this notion when we remove some unnecessary transitions from the induced DSAs that are obtained.

Definition 5.11 (Well-formed DSA)**.** A DSA $\mathcal{A}$ is *well-formed* if for every state q, no outgoing label $\alpha \in \mathrm{Out}(q)$ is a suffix of some proper prefix β' of another outgoing label $\beta \in \mathrm{Out}(q)$.

5.2. **Removing some redundant transitions.** Let us now get back to Figure 5 to see if we can derive the DSA on the left from the DFA on the right (assuming q is the initial state). As seen earlier, the set $S = \{q, q'\}$ is suffix tracking. It is also well formed since *baaa* is not a suffix of any prefix of *abaa* and vice-versa. The DSA $\mathcal{A}_S$ induced using q and q' will have the set of words in $\mathsf{SP}(q \rightsquigarrow q', S)$ as transitions between q and q'. Both *abaa* and *baaa* belong to $\mathsf{SP}(q \rightsquigarrow q', S)$. However, there are some additional simple words: for

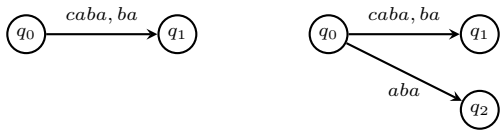

Figure 11: Illustrating bigger-suffix transitions and when they are redundant

instance, *abbaaa*. Notice that *baaa* is a suffix of *abbaaa*, and therefore even if we remove the transition on *abbaaa*, there will be a move to q' via $q \xrightarrow{baaa} q'$. This tempts us to use only the suffix-minimal words in the transitions of the induced DSA. This is not always safe, as we explain below. We show how to carefully remove "bigger-suffix-transitions".

Consider the DSA on the left in Figure 11. If *caba* is removed, the moves which were using *caba* can now be replaced by *ba* and we still have the same pair of source and target states. Consider the picture on the right of the same figure. There is an outgoing edge to a different state on *aba*. Suppose we remove *caba*. The word *caba* would then be matched by the longer suffix *aba* and move to a different state. Another kind of redundant transitions are some of the self-loops on DSAs. In Figure 8, the self-loop on b at q_0 can be removed, without changing the language. This can be generalized to loops over longer words, under some conditions.

Definition 5.12. Let $\mathcal{A}$ be a DSA, q, q' be states of $\mathcal{A}$ and $t := q \xrightarrow{\alpha} q'$ be a transition.

We call t a *bigger-suffix-transition* if there exists another transition (q, β, q') with β a suffix of α.

If there is a transition $t' := q \xrightarrow{\gamma} q''$ $(q'' \neq q')$, such that β is a suffix of γ, and γ is a suffix of α, we call t *useful*. A bigger-suffix-transition is called *redundant* if it is not useful.

We will say that t is a *redundant self-loop* if $q = q'$, q is not an accepting state, and no suffix of α is a prefix of some outgoing label in $\mathrm{Out}(q)$.

In Figure 11, for the automaton on the left, the transition on *caba* is redundant. Whereas for the DSA on the right, *caba* is a bigger-suffix-transition, but it is useful. The self-loop on q_0 in Figure 8 is redundant, but the loop on q_0 in Figure 4 is useful. Lemmas 5.13 and 5.14 prove correctness of removing redundant transitions.

Lemma 5.13. *Let $\mathcal{A}$ be a DSA, and let $t := q \xrightarrow{\alpha} q'$ be a redundant bigger-suffix-transition. Let $\mathcal{A}'$ be the DSA obtained by removing t from $\mathcal{A}$. Then, $L(\mathcal{A}) = L(\mathcal{A}')$.*

Proof. To show $L(\mathcal{A}) \subseteq L(\mathcal{A}')$. Let $w \in L(\mathcal{A})$ and let $q_0 \xrightarrow[\alpha_0]{w_0} q_1 \xrightarrow[\alpha_1]{w_1} \cdots \xrightarrow[\alpha_{m-1}]{w_{m-1}} q_m$ be an accepting run. If no (q_i, α_i, q_{i+1}) equals (q, α, q'), then the same run is present in S', and hence $w \in L(S')$. Suppose $(q_j, \alpha_j, q_{j+1}) = (q, \alpha, q')$ for some j. So, the word w_j ends with α. As (q, α, q') is a bigger-suffix-transition, there is another (q, β, q') such that $\beta \sqsubseteq_{\mathsf{sf}} \alpha$. Therefore, the word w_j also ends with β. Since there was no transition matching a proper prefix of w_j, the same will be true at $\mathcal{A}'$ as well, since it has fewer transitions. It remains to show that $q_j \xrightarrow[\beta]{w_j} q_{j+1}$ is a move. The only way this cannot happen is if there is a $q \xrightarrow{\gamma} q''$ with $\beta \sqsubseteq_{\mathsf{sf}} \gamma \sqsubseteq_{\mathsf{sf}} \alpha$. But this is not possible since $q \xrightarrow{\alpha} q'$ is a redundant bigger-suffix transition. Therefore, every move using (q, α, q') in $\mathcal{A}$ will now be replaced by (q, β, q') in $\mathcal{A}'$. Hence we get an accepting run in $\mathcal{A}'$, implying $w \in L(\mathcal{A}')$.

To show $L(\mathcal{A}') \subseteq L(\mathcal{A})$. Consider $w \in L(\mathcal{A}')$ and an accepting run $q_0 \xrightarrow[\alpha_0]{w_0} q_1 \xrightarrow[\alpha_1]{w_1} \cdots \xrightarrow[\alpha_{m-1}]{w_{m-1}} w_m$ in $\mathcal{A}'$. Notice that if $q \xrightarrow[\beta]{w_j} q'$ is a move in $\mathcal{A}'$, the same is a move in $\mathcal{A}$ when $\alpha \not\sqsubseteq_{\mathsf{sf}} w_j$. When $\alpha \sqsubseteq_{\mathsf{sf}} w_j$, then the bigger-suffix-transition $q \xrightarrow{\alpha} q'$ will match and the move $q \xrightarrow[\beta]{w_j} q'$ gets replaced by $q \xrightarrow[\alpha]{w_j} q'$. Hence we will get the same run, except that some of the moves using $q \xrightarrow{\beta} q'$ may get replaced with $q \xrightarrow{\alpha} q'$. □

For the correctness of removing redundant self-loops, we assume that the DFA that we obtain is well-formed (Definition 5.11) and has no redundant bigger-suffix-transitions. The induced DSA that we obtain from suffix-tracking sets is indeed well-formed. Starting from this induced DSA, we can first remove all redundant bigger-suffix-transitions, and then remove the redundant self-loops.

Lemma 5.14. *Let $\mathcal{A}$ be a well-formed DSA that has no removable bigger-suffix-transitions. Let $t := (q, \alpha, q)$ be a removable self-loop. Then the DSA $\mathcal{A}'$ obtained by removing t from $\mathcal{A}$ satisfies $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.*

Proof. To show $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}')$. Let $w \in \mathcal{L}(\mathcal{A})$ and let $\rho := q_0 \xrightarrow{w_0} q_1 \xrightarrow{w_1} \cdots \xrightarrow{w_{m-1}} q_m$ be an accepting run. Suppose t matches the segment $q_j \xrightarrow{w_j} q_{j+1}$. Hence $q_j = q_{j+1} = q$. Observe that as q is not accepting, we have $j + 1 \neq m$. Therefore there is a segment $q_{j+1} \xrightarrow{w_{j+1}} q_{j+2}$ in the run. We claim that if t is removed, then no transition out of q can match any prefix of $w_j w_{j+1}$.

First we see that no prefix of w_j can be matched, including w_j itself: if at all there is a match, it should be at w_j, and a β that is smaller than α. By assumption, α is not a removable bigger-suffix-transition. Therefore, there is a transition $q \xrightarrow{\gamma} q'$, with $\beta \sqsubseteq_{\mathsf{sf}} \gamma \sqsubseteq_{\mathsf{sf}} \alpha$. This contradicts the assumption that α is a removable self-loop. Therefore there is no match upto w_j.

Suppose some (q, β, q') matches a prefix $w_j u$ such that $\beta = vu$, that is, β overlaps both w_j and w_{j+1}. If $\alpha \sqsubseteq_{\mathsf{sf}} v$, then it violates well-formedness of S since it would be a suffix of a proper prefix (v) of β. This shows $v \sqsubseteq_{\mathsf{sf}} \alpha$ (since both are suffixes of w_j) and $v \sqsubseteq_{\mathsf{pr}} \beta$, contradicting the assumption that t is removable. Therefore, β does not overlap w_j. But then, if β is a suffix of a proper prefix of w_{j+1}, we would not have the segment $q_{j+1} \xrightarrow{w_{j+1}} q_{j+2}$ in the run ρ. Therefore, the only possibility is that we have a segment $q_j \xrightarrow{w_j w_{j+1}} q_{j+2}$. We have fewer occurrences of the removable loop (q, α, q) in the modified run. Repeating this argument for every match of (q, α, q) gives an accepting run of $\mathcal{A}'$. Hence $w \in L(\mathcal{A}')$.

To show $L(\mathcal{A}') \subseteq L(\mathcal{A})$. Let $w \in L(\mathcal{A}')$ and $\rho' := q_0 \xrightarrow{w_0} q_1 \xrightarrow{w_1} \cdots \xrightarrow{w_{m-1}} q_m$ be an accepting run in $\mathcal{A}'$. Suppose $q_j \xrightarrow{w_j} q_{j+1}$ is matched by (q, β, q'). Let $w_j = vu$ with $\alpha \sqsubseteq_{\mathsf{sf}} v$. Then the removable-self-loop (q, α, q) will match the prefix v. Suppose β overlaps with both v and u, that is $\beta = \beta' u$. We cannot have $\alpha \sqsubseteq_{\mathsf{sf}} \beta'$ due to well-formedness of $\mathcal{A}$. We cannot have $\beta' \sqsubseteq_{\mathsf{sf}} \alpha$ since this would mean there is a suffix of α which is a prefix of β, violating the removable-self-loop condition. Therefore, β is entirely inside u, that is, $\beta \sqsubseteq_{\mathsf{sf}} u$. Hence in $\mathcal{A}$ the run will first start with $q \xrightarrow{v} q$. Applying the same argument, prefixes of the remaining word where t matches will be matched until there is a part of the word where (q, β, q') matches. This applies to every segment, thereby giving us a run in $\mathcal{A}$. □

We now get to the core definition of this section, which tells how to derive a DSA from a DFA, using the methods developed so far.

Definition 5.15 (DFA-to-DSA derivation)**.** A DSA is said to be *derived from* DFA M using $S \subseteq Q$, if it is identical to an induced DSA of M (using S) with all redundant transitions removed.

By Theorem 5.10 and Lemma 5.13, we get the following result.

Theorem 5.16. *Every DSA that is derived from a complete DFA using a suffix-tracking set, is language equivalent to it.*

6. Minimality, some observations and some challenges

Theorem 4.5 shows that we cannot expect DSAs to be smaller than (trim) DFAs or DGAs in general. However, Lemma 4.4 and Figure 1 show that there are cases where DSAs are smaller and more readable. This motivates us to ask the question of how we can find a minimal DSA, that is, a DSA of the smallest (total) size. The first observation is that minimal DSAs need not be unique — see Figure 12.

Lemma 6.1. *The minimal DSA need not be unique.*

Proof. Figure 12 illustrates two DSAs, both of size 8 (3 states, 2 edges, total label length 3), accepting the same language. From DSA $\mathcal{A}_1$, we can see that the language accepted is $b^*a^*abb^*a$. From DSA $\mathcal{A}_2$, we can derive the expression $b^*aa^*b^*ba$. It is easy to see that both the expressions are equivalent.

It remains to show that there is no DSA of smaller size for this language. Let $L = b^*a^*abb^*a = b^*aa^*b^*ba$. Let $\mathcal{A}$ be a minimal automaton for L. Any DSA for L has an initial state q_0 which is non-accepting (since $\varepsilon \notin L$, and an accepting state $q_1 \neq q_0$). So, in particular $\mathcal{A}$ has at least two states q_0, q_1. The smallest word in L is aba. Suppose the accepting run of $\mathcal{A}$ on aba is due to a transition $t := q_0 \xrightarrow{aba} q_1$. Consider the word $abba \in L$. Transition t does not match $abba$. Therefore, the first transition in the accepting run of $\mathcal{A}$ on $abba$ is some transition $t' \neq t$. This transition t' will have a label of size at least 1. Hence, $\mathcal{A}$ has at least two states, and at least two transitions: t which contributes to size 4 (includes 1 edge and label of length 3) and a transition t' of size at least 2. Therefore $|\mathcal{A}| \geq 2 + 4 + 2 = 8$, which is at least the size of $|\mathcal{A}_1|$ and $|\mathcal{A}_2|$, and hence any DSA that accepts aba through a single transition has total-size at least 8. Now, suppose the accepting run of $\mathcal{A}$ on aba has a first transition on a prefix of aba, either a or ab, corresponding to an intermediate transition $q_0 \xrightarrow{a} q'$ or $q_0 \xrightarrow{ab} q'$. In the former case, there can be a transition $q' \xrightarrow{ba} q_1$ to accept aba (if the transition is on just b, it only makes the automaton larger since a remains to be read), and in the latter case, a transition $q' \xrightarrow{a} q_1$. This discussion shows that the two automata $\mathcal{A}_2$ and $\mathcal{A}_1$, are indeed minimal. □

The next observation is that a minimal DSAs will be well-formed: if there are two transitions $q \xrightarrow{\alpha} q_1$ and $q \xrightarrow{\beta_1\alpha\beta_2} q_2$, then we can remove the second transition since it will never get fired.

Due to the "well-formedness" property in suffix-tracking sets, the DSAs induced by suffix-tracking sets are naturally well-formed. Furthermore, since removing redundant transitions preserves this property, the DSAs that are derived using our DFA-to-DSA

Figure 12: Minimal DSA is not unique

procedure (Definition 5.15) are well-formed. The next proposition shows that every DSA that is well-formed and has no redundant transitions (and in particular, the minimal DSAs) can be derived from the corresponding tracking DFAs.

Proposition 6.2. *Every well-formed DSA with no redundant transitions can be derived from its tracking DFA.*

Proof. We use notation as in Definition 4.2. Consider a DSA $\mathcal{A}$ that is well-formed and has no redundant bigger-suffix-transitions. Let $M_{\mathcal{A}}$ be its tracking DFA as in Definition 4.1. Let $S = \bigcup_{q \in \mathcal{A}} (q, \varepsilon)$. Here is the schema of the proof:

$$M_{\mathcal{A}} \xrightarrow{S} \text{ induced DSA } \mathcal{A}' \xrightarrow{\text{remove redundant transitions}} \mathcal{A}$$

(1) We first show that S is suffix-tracking and well-formed.
(2) For each state q of $\mathcal{A}$, and for each $\alpha \in \mathsf{SP}((q, \varepsilon), S)$, either $\alpha \in \mathrm{Out}(q)$ (a transition out of q in $\mathcal{A}$) or α is a redundant bigger-suffix-transition that will be removed in the second step.

All together, this shows that $\mathcal{A}$ is derived from $M_{\mathcal{A}}$ using our procedure.

S is suffix-tracking. Pick a state (q, β) with $\beta \neq \varepsilon$. What is the set $\mathsf{SP}((q, \varepsilon) \rightsquigarrow (q, \beta), S)$? Clearly, $\beta \in \mathsf{SP}((q, \varepsilon) \rightsquigarrow (q, \beta), S)$. Let $\alpha \in \mathsf{SP}((q, \varepsilon) \rightsquigarrow (q, \beta), S)$, with $\alpha \neq \beta$. Then there is a path $(q, \varepsilon) \xrightarrow{\alpha_1} (q, \beta_1) \xrightarrow{\alpha_2} (q, \beta_2) \cdots (q, \beta_{k-1}) \xrightarrow{\alpha_k} (q, \beta_k) = (q, \beta)$. Let $j \in \{1, \ldots, k\}$ be the largest index such that $\beta_j, \beta_{j+1}, \ldots, \beta_k$ are all prefixes of β. Therefore, α starts by visiting a branch different from the prefixes of β, moves to potentially different "branches" (prefixes of other words in $\mathrm{Out}(q)$, and on $\alpha_1 \ldots \alpha_j$ hits the β branch, after which it stays in the same branch. By definition, β_j is the longest prefix among $\overline{\mathrm{Out}}(q)$ such that $\beta_j \sqsubseteq_{\mathsf{sf}} \alpha_1 \ldots \alpha_j$. We can infer the following: (1) $\beta \sqsubseteq_{\mathsf{sf}} \alpha$, and (2) among $\overline{\mathrm{Out}}(q)$, β is the longest suffix of α: otherwise, for $\alpha_1 \ldots \alpha_j$, there would be a $\beta' \neq \beta_j$ which is a longer suffix, contradicting the definition of the transition $(q, \beta_{j-1}) \xrightarrow{\alpha_j} (q, \beta_j)$.

These observations are helpful to show that S is suffix-tracking. Consider a transition $(q, \beta) \xrightarrow{a} (q, \beta')$ with $\beta' \neq \varepsilon$. We require to show that for every $\alpha \in \mathsf{SP}((q, \varepsilon) \rightsquigarrow (q, \beta), S)$, the longest suffix of αa among $\mathsf{SP}((q, \varepsilon), S)$ lies in $\mathsf{SP}((q, \varepsilon) \rightsquigarrow (q, \beta'), S)$. Pick $\alpha \in \mathsf{SP}((q, \varepsilon) \rightsquigarrow (q, \beta), S)$, and consider the extension αa. By (1) above, we have $\beta \sqsubseteq_{\mathsf{sf}} \alpha$. Since $\beta' \sqsubseteq_{\mathsf{sf}} \beta a$, we have $\beta' \sqsubseteq_{\mathsf{sf}} \alpha a$. Suppose there is an $\alpha'' \in \mathsf{SP}((q, \varepsilon) \rightsquigarrow (q, \beta''), S)$ such that $|\alpha''| > |\beta'|$ and $\alpha'' \sqsubseteq_{\mathsf{sf}} \alpha a$. By definition of the transition $(q, \beta) \xrightarrow{a} (q, \beta')$, β' is the longest suffix of βa, and hence $|\beta'| > |\beta''|$. From $|\alpha''| > |\beta'|$, $\alpha'' \sqsubseteq_{\mathsf{sf}} \alpha a$ and $\beta' \sqsubseteq_{\mathsf{sf}} \beta a$, we have $\beta' \sqsubseteq_{\mathsf{sf}} \alpha''$. By point (2) of the above paragraph, we have $|\beta''| > |\beta'|$. A contradiction. Therefore, there is no such α''. Hence β' is indeed the longest suffix of αa for all $\alpha \in \mathsf{SP}((q, \varepsilon) \rightsquigarrow (q, \beta), S)$.

Now, consider a transition $(q, \beta) \xrightarrow{a} (q, \overline{\varepsilon})$. This happens when no non-empty word in $\overline{\mathrm{Out}}(q)$ is a suffix of βa. In that case, there is a transition $(q, \varepsilon) \xrightarrow{a} (q, \overline{\varepsilon})$, and a is indeed the longest suffix of βa among $\mathsf{SP}((q, \varepsilon), S)$. This shows that every transition is suffix-compatible.

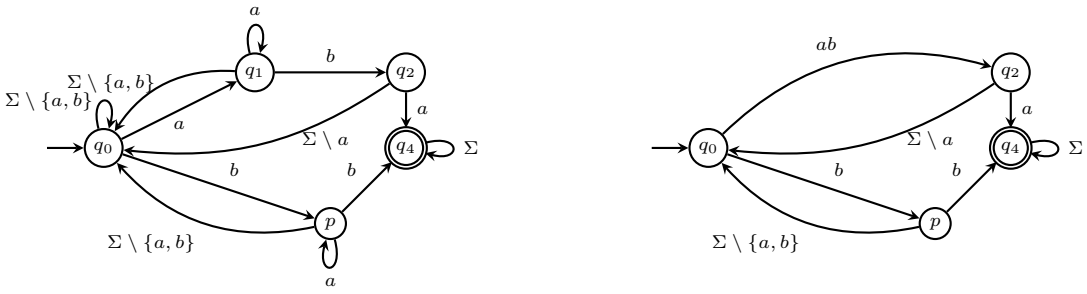

Figure 13: DFA M^* on the left and a derived DSA $\mathcal{A}^*_S$ with $S = \{q_0, q_2, q_4, p\}$ on the right.

S is well-formed. S is well-formed naturally since $\mathcal{A}$ is well-formed. Suppose it was not. That means there exist states $p \in S, q' \notin S, q$, and words $\alpha \in \mathsf{SP}(p \rightsquigarrow q, S), \beta' \in \mathsf{SP}(p \rightsquigarrow q', S)$, such that $\alpha \sqsubseteq_{\mathsf{sf}} \beta'$. Then there is a state $q'' \in S$, and a word $\beta \in \mathsf{SP}(p \rightsquigarrow q'', S)$ such that $\beta' \sqsubset_{\mathsf{pr}} \beta$, and we have $\alpha \sqsubseteq_{\mathsf{sf}} \beta'$. Since $\alpha, \beta \in \mathrm{Out}(p)$, this means $\mathcal{A}$ is not well-formed, which is a contradiction.

Extra strings in the induced DSA are redundant. In the induced DSA, we will have $\mathsf{SP}((q, \varepsilon), S)$ to have more words than $\mathrm{Out}(q)$. We need to show that all the other words are redundant bigger-suffix-transitions, and hence will be removed by the derivation procedure. Consider a transition of the form $(q, \beta) \xrightarrow{a} (q', \varepsilon)$. For every word $\alpha \in \mathsf{SP}((q, \varepsilon) \rightsquigarrow (q, \beta), S)$, with $|\alpha| > |\beta|$, we have β as the longest suffix of α, among $\overline{\mathrm{Out}}(q)$. Therefore, there is no β' with $|\alpha| \geq |\beta'| > |\beta|$ with $\beta' \sqsubseteq_{\mathsf{sf}} \alpha$. This is sufficient to see that there is no $\beta' a \in \mathrm{Out}(q)$ such that $\beta a \sqsubseteq_{\mathsf{sf}} \beta' a \sqsubseteq_{\mathsf{sf}} \alpha a$. Hence αa is a redundant bigger-suffix-transition in the induced DSA.

By assumption, we have no redundant bigger-suffix-transitions in $\mathcal{A}$. Hence, in the derivation procedure, we do not remove any transition already present in $\mathcal{A}$. This shows that the finally derived DSA is exactly $\mathcal{A}$. □

Proposition 6.2 says that if we somehow had access to the tracking DFA of a minimal DSA, we will be able to derive it using our procedure. The challenge however is that this tracking DFA may not necessarily be the canonical DFA for the language. In fact, we now show that a smallest DSA that can be derived from the canonical DFA need not be a minimal DSA.

Figure 13 shows a DFA M^*. Observe that M^* is minimal: every pair of states has a distinguishing suffix. Let us now look at DSAs that can be derived from M^*. Firstly, any suffix-tracking set on M^* would contain q_0, q_4 (since they are initial and accepting states). If p is not picked, the transition $p \xrightarrow{a} p$ is not suffix-compatible. Therefore, p should belong to the selected set. If p is picked, and q_2 not picked, then the set is not well-formed (see Definition 5.8): the simple word b from q_0 to p is a suffix of the simple word ab to q_2. Therefore, any suffix-tracking set should contain the 4 states q_0, p, q_2, q_4. This set $S = \{q_0, p, q_2, q_4\}$ is indeed suffix-tracking, and the DSA derived using S is shown in the right of Figure 13. The only other suffix-tracking set is the set S' of all states. The DSA derived using S' will have state q_1 in addition, and the transitions $\Sigma \setminus \{a, b\}$. If Σ is sufficiently large, this DSA would have total size bigger than $\mathcal{A}^*_S$ (Note: Σ can be any alphabet containing a

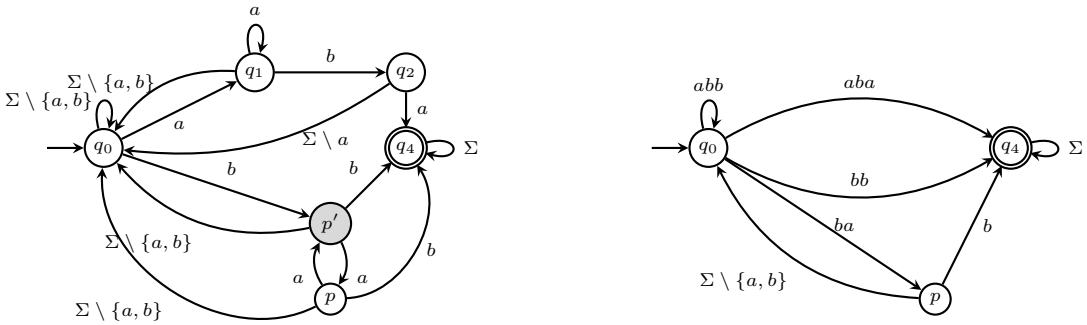

Figure 14: DFA M^{**} on the left and a derived DSA $\mathcal{A}_S^{**}$ with $S = \{q_0, q_2, q_4, p\}$ on the right.

and b, so we can increase its size arbitrarily to blow up the DSA). We deduce $\mathcal{A}_S^*$ to be the smallest DSA that can be derived from M^*.

Figure 14 shows DFA M^{**} which is obtained from M^* by duplicating state p to create a new state p', which is equivalent to p. So M^{**} is language equivalent to M^*, but it is not minimal. Here, if we choose p in a suffix-tracking set, the simple word to p is ba, which is not a suffix of ab (the simple word to q_2). Hence, we are not required to add q_2 into the set. Notice that $S = \{q_0, p, q_4\}$ is indeed a suffix-tracking set in M^{**}. The derived DSA $\mathcal{A}_S^{**}$ is shown in the right of the figure. The "heavy" transition on $\Sigma \setminus a$ disappears. There are some extra transitions, like $q_0 \xrightarrow{bb} q_4$, but if Σ is large enough, the size of $\mathcal{A}_S^{**}$ will be smaller than $\mathcal{A}_S^*$. This shows that starting from a big DFA helps deriving a smaller DSA, and in particular, the canonical DFA of a regular language may not derive a minimal DSA for the language.

7. Strongly Deterministic Suffix-reading Automata (sDSA)

As we saw in Section 6, the minimal DFA of a regular language may not derive a minimal DSA for the language (Figures 13 and Figure 14). While this is true for the general case, under a restricted definition of DSA we can show the minimal DFA to derive a minimal (restricted) DSA. The plan for this section is as follows:

Section 7.1: We first define a class of DSAs with a restricted syntax and call them *strongly deterministic* suffix-reading automata or *strong DSAs* (Definition 7.1) and give examples of DSAs that fall under this stronger class.

Section 7.2: In the second part, we prove that every minimal strong DSA can be derived from the canonical DFA using the derivation procedure of Section 5.

This section is inspired by a question that was left open in the earlier version of this work [KSVV24]: when does the smallest DSA derived from the canonical DFA correspond to a minimal DSA? Although we do not provide an answer to this question, we now understand that when we suitably restrict the DSA syntax, the derivation procedure indeed is able to generate a minimal automaton in the restricted class, starting from the canonical DFA. This provides new insights about the derivation procedure. Moreover, as we will see, many DSAs discussed in this paper are already strong. We also provide a real-life-inspired example of a strong DSA. So, overall, this section sends the message that there are specifications that can

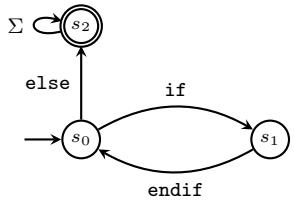

Figure 15: sDSA for out-of-context `else`, without nested if statements

be encoded naturally as strong DSAs, and we have a method to generate minimal strong DSAs.

7.1. **Syntax of strong DSAs.** We start with a formal description of the syntax of strong DSAs. The examples that follow aim to give an explanation of this definition. In this definition, we say α' is a non-empty prefix of a word α when $\alpha' \neq \varepsilon$ and $\alpha' \sqsubseteq_{\mathsf{pr}} \alpha$.

Definition 7.1 (sDSA)**.** (Strongly Deterministic Suffix-reading Automata). A DSA $\mathcal{A} = (Q, \Sigma, q_{init}, \Delta, F)$ is said to be strongly deterministic if for every state $q \in Q$, for every $\alpha, \beta \in \mathrm{Out}(q)$, and for all non-empty prefixes $\alpha' \sqsubseteq_{\mathsf{pr}} \alpha$ and all non-empty proper prefixes $\beta' \sqsubset_{\mathsf{pr}} \beta$, we have $\alpha' \not\sqsubseteq_{\mathsf{sf}} \beta'$: no prefix of α is a factor of β.

For instance, the DSA in Figure 1 is not an sDSA: at state s_1 we have $\mathrm{Out}(s_1) = \{\texttt{if}, \texttt{endif}\}$, and the letter `i` (which is a prefix of `if`) appears in `endif` as a suffix of `endi`. The DSAs in Figures 3 and 4 are strong DSAs: it is easy to verify this for $\mathcal{A}_2$ and $\mathcal{A}_3$; for $\mathcal{A}_4$, we provide an explanation. We have $\mathrm{Out}(q_0) = \{ab, ba\}$; let $\alpha = ab$ and $\beta = ba$. The non-empty prefixes of α are a and ab. The only non-empty proper prefix of β is b. Notice that neither a nor ab is a suffix of b. The same exercise can be repeated with $\alpha = ba$ and $\beta = ab$. We conclude that $\mathcal{A}_4$ is an sDSA. Note that any DFA is an sDSA, as the labels of each outgoing transition at a state are distinct letters (and a DFA is a valid DSA). A slightly modified version of the DSA in Figure 1 gives us an example of an sDSA. In Figure 15 we have a DSA for out-of-context `else` statements, where we allow nested `if`, but a single `endif` appearing later corresponds to all the open `if` so far. A context would therefore be the part between the first `if` and the last `endif`. If the specification additionally requires to disallow nested `if`, then it can be modeled by the intersection of the sDSA of Figure 15 and another automaton that rejects words with two `if` with no `endif` in between.

We describe another example of a strong DSA, motivated by a real-life situation. It is a simplified specification of an automotive application: "when the alarm is off, and the panic switch is pressed twice within one clock cycle, go to an error state". The DFA in Figure 16 models this. States q_0 and q_1 represent the alarm being off and on respectively. State q_3 is the error state. The DFA toggles between off and on states on receiving Signal s. Signal t denotes a "tick" marking the separation of clock cycles, and p denotes the pressing of the panic switch. The sDSA for this specification is given on the right of Figure 16. At state q_0, the automaton keeps receiving signals until either an s or a sequence pp is seen. If s is seen first, the automaton moves to q_1, otherwise it moves to q_3. The move on $q_0 \xrightarrow{pp} q_3$ happens on words over $\{t, p, s\}$ that end with pp, and contain neither an s nor a previous occurrence of pp. This expresses that the panic switch was pressed twice, within one clock cycle (as no t occurred), while the alarm was still off. Here, the sDSA provides a smaller, and arguably, a more readable representation than the DFA.

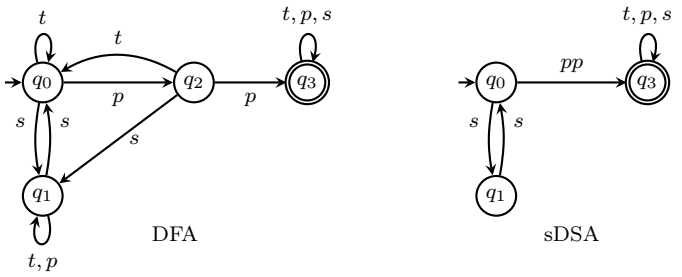

Figure 16: A simplified specification of an automotive application

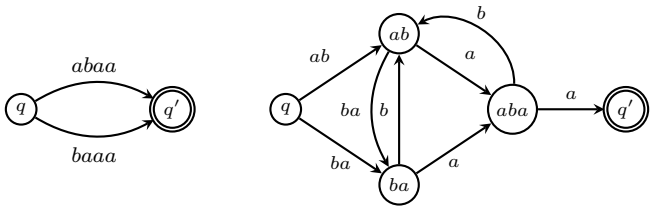

Figure 17: A DSA on the left, and the corresponding (larger) sDSA

However sDSA may not always provide as succinct a representation as DSA. We see in Figure 17, an example of a DSA on the left and an equivalent sDSA on the right. It is the DSA from Figure 5 again, with the corresponding sDSA shown. The DSA itself is not a valid sDSA, since ba is a prefix of $baaa$ and a factor of $abaa$ (also a is prefix of $abaa$ and a factor of $baaa$). Intuitively, to construct an sDSA, we need to break the transitions after ab and ba and introduce new states. This continues to happen on the next transitions from these new states resulting in the sDSA shown.

7.2. **Minimality for strong DSAs.** We state some preliminary lemmas, leading up to the main result that any minimal sDSA is derived from a minimal DFA. The next lemma is generic and holds for all DSAs, which are not necessarily strong. We start with some notation.

For a state q of a DFA M, we define $L^M(q)$ to be the words accepted by M starting from q and call it the *residual language* of the state. Similarly, for a DSA $\mathcal{A}$ and a state q of $\mathcal{A}$, we can define the residual language $L^{\mathcal{A}}(q)$. We say that two states p, q of a DFA/DSA are *equivalent* if their residual languages are equal. In the setting of DFAs, we know that in the canonical (minimal-state) automaton, no two states are equivalent. The next lemma establishes this property even in the DSA setting.

Lemma 7.2. *No two states of a minimal DSA can be equivalent.*

Proof. Let $\mathcal{A}$ be a minimal DSA. Suppose $L^{\mathcal{A}}(p) = L^{\mathcal{A}}(q), p \neq q$. We will now construct a DSA $\mathcal{A}'$ with smaller size than $\mathcal{A}$. This will be a contradiction. To get $\mathcal{A}'$, we will re-orient all transitions going to q to now point towards p: remove each transition (r, α, q) and add the transition (r, α, p); then remove q and other unreachable states after this transformation. Since $L^{\mathcal{A}}(p) = L^{\mathcal{A}}(q)$, this construction preserves the language. For all the states r that remain in $\mathcal{A}'$ we still have the same $\mathrm{Out}(r)$ as in $\mathcal{A}$. Finally we need to argue that $|\mathcal{A}'| < |\mathcal{A}|$.

This is easy to see since q has been removed from $\mathcal{A}$, and there are no new additions to $\mathcal{A}'$. □

Now we come to properties specific to sDSAs. The key idea is to consider the tracking DFA (Definition 4.1 and Figure 5) and investigate equivalence between states in this tracking DFA.

Lemma 7.3. *For any minimal sDSA $\mathcal{A}$, its tracking DFA $M_\mathcal{A}$ cannot have a 'DSA state' equivalent to any other state, i.e. if $(p,\alpha),(q,\beta)$ are states of $M_\mathcal{A}$ and also equivalent, then we have both $\alpha \neq \varepsilon$ and $\beta \neq \varepsilon$.*

Proof. From Lemma 7.2, we know that no two states of any minimal DSA can be equivalent. So it is not possible to have any distinct $(p,\varepsilon),(q,\varepsilon) \in M_\mathcal{A}$ to be equivalent. Suppose $L^{M_\mathcal{A}}(p,\varepsilon) = L^{M_\mathcal{A}}(q,\beta)$ for some $\beta \neq \varepsilon$ (and q could be p as well). We will construct a smaller sDSA $\mathcal{A}'$.

Consider state q of $\mathcal{A}$. Due to the presence of state (q,β) in $M_\mathcal{A}$, there exists some outgoing label of the form $\beta\alpha$ from q in $\mathcal{A}$: that is, $\beta\alpha \in \mathrm{Out}(q)$ for some α with $|\alpha| \geq 1$. Let the corresponding transition be $q \xrightarrow{\beta\alpha} r$. To get $\mathcal{A}'$: remove this transition $q \xrightarrow{\beta\alpha} r$ and add the transition $q \xrightarrow{\beta} p$. Clearly $\mathcal{A}'$ is smaller than $\mathcal{A}$, since total size is reduced by $|\alpha|$. Language accepted is the same. $\mathcal{A}'$ is also an sDSA since the only change is replacing $\beta\alpha$ with β in $\mathrm{Out}(q)$: any string in $\overline{\mathrm{Out}}(q)$ that is a factor of β would also be a factor of $\beta\alpha$, and any prefix of β is also a prefix of $\beta\alpha$. This contradicts the assumption that $\mathcal{A}$ was a minimal sDSA, thus proving the lemma. □

We remark that the above proof does not work for general DSAs. Consider the transition $q \xrightarrow{\beta\alpha} r$ that was removed, and modified to $q \xrightarrow{\beta} p$. There could be another transition $q \xrightarrow{\beta\alpha'} r'$ in the DSA — this violates the strong DSA property since $\beta\alpha$ and $\beta\alpha'$ are outgoing labels, and the prefix β of the former appears as a suffix of the latter. On seeing the word $\beta\alpha'$ from q, the original DSA moves to r'. However, in the new DSA, as soon as α is seen the state changes to p. This can modify the language. Such a situation does not happen in an sDSA due to the restriction on the outgoing labels. We now prove the main result of this section.

Theorem 7.4. *Every minimal sDSA for a language L can be derived from the canonical DFA for L.*

Proof. Let $\mathcal{A} = (Q^\mathcal{A}, \Sigma, \Delta^\mathcal{A}, F^\mathcal{A})$ be an arbitrary minimal sDSA. Any minimal sDSA is well-formed and has no redundant transitions. Hence $\mathcal{A}$ can be derived from its tracking DFA $M_\mathcal{A}$, thanks to Proposition 6.2. If the tracking DFA $M_\mathcal{A}$ is canonical, we are done. Otherwise, there are distinct states of $M_\mathcal{A}$ that are equivalent. From Lemma 7.3, we know that a state (q,ε) of $M_\mathcal{A}$ cannot be equivalent to any other state. Therefore, two equivalent states are of the form (p,α) and (q,β), with α and β both non-empty. We will merge such equivalent states together to build the minimal automaton and make use of this specific structure to show that $\mathcal{A}$ can be derived from the minimized DFA. For the proof, we will show the following steps.

(1) Suppose a state (q,α) is equivalent to (q,β) (with the same q), then $\alpha \not\sqsubseteq_{\mathsf{pr}} \beta$ (i.e. the two states cannot be 'tracking' the same $\mathcal{A}$-transition out of q).
(2) Build a DFA $M'_\mathcal{A}$ by quotienting equivalent states of $M_\mathcal{A}$, with $[q]$ representing the equivalence class of q. The resulting automaton $M'_\mathcal{A}$ is the canonical DFA. We use (1) to show that every simple path from a state (q,ε) to (q,β) is preserved in the quotiented automaton $M'_\mathcal{A}$, from $[(q,\varepsilon)]$ to $[(q,\beta)]$.

(3) From (2), we show that the set of states $S' := \{[(q,\varepsilon)] \mid q \in Q^{\mathcal{A}}\}$ forms a suffix-tracking set. The DSA derived using S' turns out to be $\mathcal{A}$, proving that $\mathcal{A}$ can be derived from the canonical DFA.

Step 1. If two states $(q,\alpha),(q,\beta) \in M_{\mathcal{A}}$ are equivalent, then $\alpha \not\sqsubseteq_{\mathsf{pr}} \beta$ (i.e. the two states are not 'tracking' the same $\mathcal{A}$-transition). In other words, two states in the tracking DFA that lie on the same path corresponding to a given DSA transition, cannot be equivalent. Suppose $\alpha \sqsubseteq_{\mathsf{pr}} \beta$. We know that $\beta \in \overline{\mathrm{Out}}(q)$, so there is a string γ such that $\beta\gamma \in \mathrm{Out}(q)$ i.e. reading γ from (q,β) leads to a 'DSA state', say (q',ε). Let s be the state reached on reading γ from (q,α). Since (q,α) and (q,β) are equivalent, the state s will be equivalent to (q',ε). From Lemma 7.3, this means $s = (q',\varepsilon)$. This gives transition labels $\alpha\gamma$ (induced by the simple path just discussed) and $\beta\gamma$ from q in $\mathcal{A}$, with $\alpha \sqsubseteq_{\mathsf{pr}} \beta$, contradicting the fact that $\mathcal{A}$ was an sDSA. Hence we must have $\alpha \not\sqsubseteq_{\mathsf{pr}} \beta$.

Step 2. For two states s, s' of $M_{\mathcal{A}}$, define $s \equiv_{M_{\mathcal{A}}} s'$ if $L^{M_{\mathcal{A}}}(s) = L^{M_{\mathcal{A}}}(s')$. We denote the equivalence class of s by $[s]$. From Lemma 7.3, $[(q,\varepsilon)] = \{(q,\varepsilon)\}$. Consider a quotient DFA $M'_{\mathcal{A}}$ based on this equivalence. States are the equivalence classes of $\equiv_{M_{\mathcal{A}}}, \{[s] \mid s \in M_{\mathcal{A}}\}$. The initial state is $[(q^{\mathcal{A}}_{in},\varepsilon)]$ and final states are $\{[(q,\varepsilon)] \mid q \in F^{\mathcal{A}}\}$. Transitions are given by $\{[s] \xrightarrow{a} [s'] \mid s \xrightarrow{a} s' \in M_{\mathcal{A}}\}$. This is deterministic because whenever we have $[s_1] = [s'_1]$ and $s_1 \xrightarrow{a} s_2, s'_1 \xrightarrow{a} s'_2 \in M_{\mathcal{A}}$, we also have $[s_2] = [s'_2]$. Note that $M'_{\mathcal{A}}$ is minimal, by definition.

Consider a transition $q \xrightarrow{\alpha} q'$ of the sDSA $\mathcal{A}$, with $\alpha = a_1 a_2 \dots a_n$. This transitions gives states (q,ε), (q,a_1), (q,a_1a_2), $\dots$, $(q,a_1\dots a_{n-1})$, (q',ε) in the tracking DFA $M_{\mathcal{A}}$. From (1), we have $[(q,a_1)],[(q,a_1a_2)],\dots,[(q,a_1a_2\dots a_{n-1})]$ to be distinct states in $M'_{\mathcal{A}}$. equivalent. Hence $a_1a_2\dots a_i$ is a simple path from $[(q,\varepsilon)]$ to $[(q,a_1\dots a_i)]$ that does not visit any state of the form $[(p,\varepsilon)]$ in between.

Step 3. Let $S' := \bigcup_{q\in\mathcal{A}}\{[(q,\varepsilon)]\}$. We will show that S' is suffix-tracking. Consider a simple path σ from $[(q,\varepsilon)]$ to $[(q,\beta)]$ (note that σ is also a simple path from (q,ε) to (q,β) in $M_{\mathcal{A}}$). In the tracking DFA $M_{\mathcal{A}}$, every transition moves to a state tracking the longest possible suffix: $(q,\beta) \xrightarrow{a} (q,\beta')$ in $M_{\mathcal{A}}$ means β' is the longest word in $\overline{\mathrm{Out}}(q)$ s.t $\beta' \sqsubseteq_{\mathsf{sf}} \sigma a$. From (2), β' is a simple path from $[(q,\varepsilon)]$ to $[(q,\beta')]$, in $M'_{\mathcal{A}}$. Moreover, we have $[(q,\beta)] \xrightarrow{a} [(q,\beta')]$ by definition.

We claim that the longest suffix of σa among simple paths from $[(q,\varepsilon)]$, is β', which goes from $[(q,\varepsilon)]$ to $[(q,\beta')]$. This would show suffix-compatibility of the transition. Suppose instead, the longest suffix (say σ') went from $[(q,\varepsilon)]$ to $[(q,\alpha')]$ in $M'_{\mathcal{A}}$; we have σ' to also be a simple path from (q,ε) to (q,α') in $M_{\mathcal{A}}$, and α' to be longer than β'. This is a contradiction. Hence β' is the longest simple-path suffix of σa in $M'_{\mathcal{A}}$, making $[(q,\beta)] \xrightarrow{a} [(q,\beta')]$ suffix-compatible.

The set S' is also well-formed since $\mathcal{A}$ is well-formed. Suppose it was not. That means there exist states $p \in S', q' \notin S'$, and words $\alpha \in \mathsf{SP}(p \rightsquigarrow q, S'), \beta' \in \mathsf{SP}(p \rightsquigarrow q', S')$, such that $\alpha \sqsubseteq_{\mathsf{sf}} \beta'$. Then there is a state $q'' \in S'$, and a word $\beta \in \mathsf{SP}(p \rightsquigarrow q'', S')$ such that $\beta' \sqsubset_{\mathsf{pr}} \beta$, and we have $\alpha \sqsubseteq_{\mathsf{sf}} \beta'$. Since $\alpha, \beta \in \mathrm{Out}(p)$ are paths corresponding to $\mathcal{A}$-transitions, this means $\mathcal{A}$ is not an sDSA, which is a contradiction. Hence we have S' to be suffix-tracking. In the induced DSA $\mathcal{A}_{S'}$, every simple path from $[(q,\varepsilon)]$ to $[(q',\varepsilon)]$ will be present. Hence every transition $q \xrightarrow{\alpha} q'$ in $\mathcal{A}$ is present in $\mathcal{A}_{S'}$. Any transition out of $[(q,\varepsilon)]$ which is not in $\mathrm{Out}(q)$ in $\mathcal{A}$, will be a redundant bigger-suffix-transition, analogous to the argument in last part of the proof of Proposition 6.2. Thus $\mathcal{A}$ can be derived from $M'_{\mathcal{A}}$, which is a minimal DFA. □

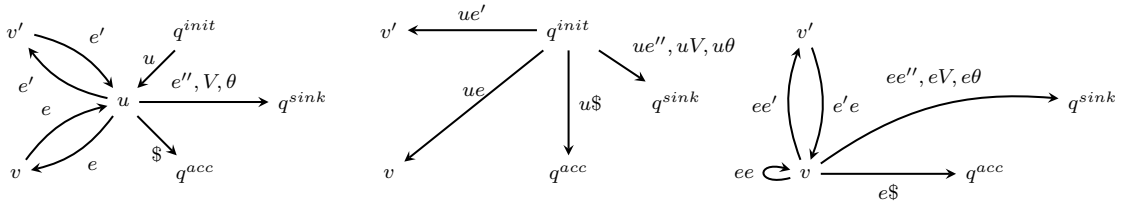

Figure 18: Left: Illustration of the neighbourhood of state u in the DFA M_G. Middle, Right: Transitions induced from q^{init} and v, on removing u.

8. Complexity of minimization

The goal of this section is to prove the following theorem.

Theorem 8.1. *Given a DFA M and positive integer k, deciding whether there exists a DSA of total size $\leq k$ language equivalent to M is NP-complete.*

If k is bigger than the size of the DFA M, then the answer is trivial. Therefore, let us assume that k is smaller than the DFA size. For the NP upper bound, we guess a DSA of total size k, compute its tracking DFA in time $\mathcal{O}(k \cdot |\Sigma|)$ and check for its language equivalence with the given DFA M. This can be done in polynomial-time by minimizing both the DFA and checking for isomorphism.

The rest of the section is devoted to proving the lower bound. We provide a reduction from the minimum vertex cover problem which is a well-known NP-complete problem [Kar72]. A vertex cover of an undirected graph $G = (V, E)$ is a subset $S \subseteq V$ of vertices, such that for every edge $e \in E$, at least one of its end points is in S. The decision problem takes a graph G and a number $k' \geq 1$ as input and asks whether there is a vertex cover of G with size at most k'. Using the graph G, we will construct a DFA M_G over an alphabet Σ_G. We then show that G has a vertex cover of size $\leq k'$ iff M_G has an equivalent DSA with total size $\leq k$ where $k = (k' + 2) \times 2\theta + (2\theta - 1)$. Here, θ is a sufficiently large polynomial in $|V|, |E|$ which we will explain later.

The alphabet Σ_G is given by $V \cup E \cup \{\$\} \cup D$ where $D = \{1, 2, \ldots, \theta\}$. States of M_G are $V \cup \{q^{init}, q^{sink}, q^{acc}\}$. For simplicity, we use the same notation for v as a vertex in G, v as a letter in Σ_G and v as a state of M_G. The actual role of v will be clear from the context. For every edge $e = (u, v)$, there are two transitions in the automaton: $u \xrightarrow{e} v$ and $v \xrightarrow{e} u$. For every $v \in V$, there are transitions $q^{init} \xrightarrow{v} v$ and $v \xrightarrow{\$} q^{acc}$. This automaton can be completed by adding all missing transitions to the sink state q^{sink}. Figure 18 (left) illustrates the neighbourhood of a state u. The notation e'' stands for any edge that is not incident on u; there is one transition for every such e''. Initial and accepting states are respectively q^{init} and q^{acc}. Let $L_G(u)$ be the set of words that have an accepting run in M_G starting from u as the initial state. If $u \neq v$, $L_G(u) = L_G(v)$ implies (u, v) is an edge and there are no other edges outgoing either from u or v. To avoid this corner case, we restrict the vertex cover problem to connected graphs of 3 or more vertices. Then we have M_G to be a minimal DFA, with no two states equivalent. Here are two main ideas.

Suppressing a state. Suppose state u of M_G is suppressed (i.e. u is not in a suffix-tracking set). In Figure 18, we show the induced transitions from q^{init} and a vertex v. However, some of them will be redundant transitions: most importantly, the set of transitions $q^{init} \xrightarrow{u1,u2,\ldots,u\theta} q^{sink}$ will be redundant bigger-suffix-transitions due to $q^{init} \xrightarrow{1,2,\ldots,\theta} q^{sink}$.

Similarly, $v \xrightarrow{e1,e2,\ldots,e\theta} q^{sink}$ will be removed. There are some more redundant bigger-suffix-transitions, like $v \xrightarrow{ee''} q^{sink}$ for some e'' that is not incident on v and u. So from each v, at most $2|E|$ transitions are added. But crucially, after removing redundant transitions, the θ transitions from u no longer appear. If we choose θ large enough to compensate for the other transitions, we get an overall reduction in size by suppressing states.

Two states connected by an edge cannot both be suppressed. Suppose $e = (u, v)$ is an edge. If S is a set where $u, v \notin S$, then the transition $v \xrightarrow{e} u$ is not suffix-compatible: the simple word ue from q^{init} to v, when extended with e gives the word uee; no suffix of uee is a simple word from q^{init} to u. We deduce that suffix-tracking sets in M_G correspond to a vertex cover in G, and vice-versa.

These two observations lead to a translation from minimum vertex cover to suffix-tracking sets with least number of states. Due to our choice of θ, DSAs with smallest (total) size are indeed obtained from suffix-tracking sets with the least number of states. Let $k = (k' + 2) \times 2\theta + (2\theta - 1)$. Below, we elaborate these ideas in more detail and present the proof of the reduction.

Vertex cover $\leq k'$ implies DSA $\leq k$. Assume there is a vertex cover $\{v_1, \ldots, v_p\}$ in G with $p \leq k'$. Let S be the set of states in M_G corresponding to $\{v_1, \ldots, v_p\}$. Observe that $S \cup \{q^{init}, q^{sink}, q^{acc}\}$ is a suffix-tracking set; every transition is trivially suffix-compatible ($\forall q \xrightarrow{a} u, q \in S$ or $u \in S$). Well-formedness holds because $\forall p, q \in S, \alpha \in \mathsf{SP}(p \rightsquigarrow q, S)$ we have $|\alpha| \leq 2$; this means $\forall q' \notin S, \beta \in \mathsf{SP}(p \rightsquigarrow q', S)$, we have $\alpha \not\sqsubseteq_{\mathsf{sf}} \beta$ (since $|\beta| = 1$). Hence the derived DSA will be equivalent to M.

The derived DSA has $p + 3$ states, and transitions $q \xrightarrow{1,2,\ldots,\theta} q^{sink}$ from each except for the q^{sink} state. The transitions on q^{sink} are removable, and hence will be absent. All of this adds $(p + 2) \times 2\theta$ to the total size (edges + label lengths). Apart from these, there are transitions with labels of length at most 2, over the alphabet $V \cup E \cup \$$. From each vertex, v, there are $|V|$ transitions to q^{sink}, one transition to q^{acc} and at most $2|E|$ transitions to other states or q^{sink}. We can choose a large enough θ (say $(|V| + |E|)^4$), so that the size of these extra transitions is at most $2\theta - 1$. Hence, total size is $\leq (p + 2) \times 2\theta + (2\theta - 1)$.

By assumption, we have $p \leq k'$. Therefore, the size of the DSA is $\leq (k'+2)\times 2\theta + (2\theta - 1) = k$.

DSA $\leq k$ implies vertex cover $\leq k'$. Let $\mathcal{A}$ be a DSA with size $\leq k$. It may not be derived from M_G. However, by Proposition 6.2 we know $\mathcal{A}$ is derived from a DFA M, the tracking DFA for $\mathcal{A}$. Moreover since M_G is the minimal DFA, we know that M will be a *refinement* of M_G (see Section 2 for definition).

Let us consider a pair of states u and v from M_G, such that the vertices $u, v \in G$ have an edge between them labeled e. The DFA M will have two sets of states $u_1, u_2, \ldots, u_i$ and $v_1, v_2, \ldots, v_j$ that are language-equivalent to u and v respectively. Its initial state must have a transition on v to one of $v_1, v_2, \ldots, v_j$. Without loss of generality, let it be to v_1. Each of $v_1, v_2, \ldots, v_j$ must have a transition on e to one of $u_1, u_2, \ldots, u_i$ (for equivalence with M_G) and vice-versa. Consider the run from the initial state on ve^{i+j+1}. At least one of the states among $u_1, u_2, \ldots, u_i, v_1, v_2, \ldots, v_j$ must be visited twice; consider the first such instance. The transition on e that re-visits a state cannot be suffix-compatible w.r.t a set S, if none of these states are in S. For it to be suffix-compatible, the string $ve^k.e$ (from initial state to the first repeated state) must have its longest simple-word suffix go to the

same state. Since $ve^k.e$ is not simple by itself, its longest suffix must consist entirely of e's. But on any string of e's, the initial state moves only to the sink state(s) and not to any of $u_1, u_2, \ldots, u_i, v_1, v_2, \ldots, v_j$. Hence any suffix-tracking set must contain at least one of these states, which maps to at least one of u or v in G. Every suffix-tracking set of M therefore maps to a vertex cover $\{v_1, v_2, \ldots, v_p\}$.

Now we show that the size of this vertex cover is $\leq k'$. Each of the states picked in the suffix-tracking set will contribute to atleast 2θ in the total size, due to the θ transitions. We will also have these θ transitions from the initial and accepting states. Therefore, the total size is $(p+2) \times 2\theta + y$ for some $y > 0$. Hence $(p+2) \times 2\theta \leq k$. This implies $p \leq k'$: otherwise we will have $p \geq k' + 1$, and hence $(p + 2) \times 2\theta \geq (k' + 1 + 2) \times 2\theta = (k' + 2) \times 2\theta + 2\theta > k$, a contradiction.

9. Conclusion

We have introduced the model of deterministic suffix-reading automata, compared its size with DFAs and DGAs, proposed a method to derive DSAs from DFAs, and presented the complexity of minimization. The work on DGAs [GM99] inspired us to look for methods to derive DSAs from DFAs, and investigate whether they lead to minimal DSAs for a language. This led to our technique of suffix-tracking sets, which derives DSAs from DFAs. The technique imposes some natural conditions on subsets of states, for them to be tracking patterns at each state. However, surprisingly, the smallest DSA that we can derive from the canonical DFA need not correspond to the minimal DSA of a language. We have shown that when restricting the syntax, our derivation method is able to generate a minimal DSA in the restricted class, starting from the canonical DFA.

In this introductory work on DSAs, our goal has been to present the model, its motivations and establish ingredients for a deeper study of the model, both from a practical and a theoretical perspective. Recently, we have enhanced the DSA syntax to include a parallel composition operator $\|$ in the transitions [KSV25]. This helps succinct representation of the patterns when the alphabet is distributed across multiple components in a concurrent system. Using the enhanced model, we have provided a formal semantics and test generation algorithm (with guarantees) for an industrial formalism called Expressive Decision Tables (EDT) [VSKA14] developed by the industry partners in this work. This work shows a direct impact of the DSA model in an industrial setting.

From a theoretical perspective, there are plenty of problems to ponder about. We do not yet have an algorithm that can start with the canonical DFA, perform some operations on it and get a minimal DSA (in the general case, and not the strong DSA case as discussed in Section 7). As we saw in Section 6, one might need to expand the canonical DFA to get the minimal DSA. It would be interesting to have a clear algorithm to identify this expansion on which the suffix-tracking techniques can be applied. Can we use our techniques to study minimality in terms of number of states? Closure properties of DSAs - can we perform the union, intersection and complementation operations on DSAs without computing the entire equivalent DFAs? What about Myhill-Nerode style congruences for DSAs? Recently a Myhill-Nerode theorem has been established for deterministic generalized automata [Cot24]. To sum up, we believe the DSA model offers advantages in the specification of systems and in also studying regular languages from a different angle. The results that we have presented throw light on some of the different aspects in this model, and lead to many questions both from theoretical and practical perspectives.

References

[BHV04] Ahmed Bouajjani, Peter Habermehl, and Tomás Vojnar. Abstract regular model checking. In *Computer Aided Verification, 16th International Conference, CAV 2004, Boston, MA, USA, July 13-17, 2004, Proceedings*, pages 372–386, 2004. doi:10.1007/978-3-540-27813-9_29.

[BJNT00] Ahmed Bouajjani, Bengt Jonsson, Marcus Nilsson, and Tayssir Touili. Regular model checking. In *Computer Aided Verification, 12th International Conference, CAV 2000, Chicago, IL, USA, July 15-19, 2000, Proceedings*, pages 403–418, 2000. doi:10.1007/10722167_31.

[BM63] Janusz A. Brzozowski and Edward J. McCluskey. Signal flow graph techniques for sequential circuit state diagrams. *IEEE Trans. Electron. Comput.*, 12(2):67–76, 1963. doi:10.1109/PGEC.1963.263416.

[CGK+18] Edmund M. Clarke, Orna Grumberg, Daniel Kroening, Doron A. Peled, and Helmut Veith. *Model checking, 2nd Edition*. MIT Press, 2018. URL: https://mitpress.mit.edu/books/model-checking-second-edition.

[Cot24] Nicola Cotumaccio. A myhill-nerode theorem for generalized automata, with applications to pattern matching and compression. In Olaf Beyersdorff, Mamadou Moustapha Kanté, Orna Kupferman, and Daniel Lokshtanov, editors, *41st International Symposium on Theoretical Aspects of Computer Science, STACS 2024, March 12-14, 2024, Clermont-Ferrand, France*, volume 289 of *LIPIcs*, pages 26:1–26:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.STACS.2024.26.

[D'A] Loris D'Antoni. Symbolic automata. https://pages.cs.wisc.edu/~loris/symbolicautomata.html.

[DV17] Loris D'Antoni and Margus Veanes. The power of symbolic automata and transducers. In *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*, pages 47–67, 2017. doi:10.1007/978-3-319-63387-9_3.

[Eil74] Samuel Eilenberg. *Automata, languages, and machines. A*. Pure and applied mathematics. Academic Press, 1974. URL: https://www.worldcat.org/oclc/310535248.

[Fer09] Henning Fernau. Algorithms for learning regular expressions from positive data. *Information and Computation*, 207(4):521–541, 2009. doi:10.1016/j.ic.2008.12.008.

[GH96] Noa Globerman and David Harel. Complexity results for two-way and multi-pebble automata and their logics. *Theor. Comput. Sci.*, 169(2):161–184, 1996. doi:10.1016/S0304-3975(96)00119-3.

[GH01] D. Giannakopoulou and K. Havelund. Automata-based verification of temporal properties on running programs. In *Proceedings 16th Annual International Conference on Automated Software Engineering (ASE 2001)*, pages 412–416, 2001. doi:10.1109/ASE.2001.989841.

[GM99] Dora Giammarresi and Rosa Montalbano. Deterministic generalized automata. *Theoretical Computer Science*, 215(1-2):191–208, 1999. doi:10.1016/S0304-3975(97)00166-7.

[Har87] David Harel. Statecharts: A visual formalism for complex systems. *Sci. Comput. Program.*, 8(3):231–274, 1987. doi:10.1016/0167-6423(87)90035-9.

[Has91] Kosaburo Hashiguchi. Algorithms for determining the smallest number of nonterminals (states) sufficient for generating (accepting) a regular language. In *Automata, Languages and Programming, 18th International Colloquium, ICALP91, Madrid, Spain, July 8-12, 1991, Proceedings*, pages 641–648, 1991. doi:10.1007/3-540-54233-7_170.

[HMU07] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation, 3rd Edition*. Pearson international edition. Addison-Wesley, 2007.

[HW04] Yo-Sub Han and Derick Wood. The generalization of generalized automata: Expression automata. In *Implementation and Application of Automata, 9th International Conference, CIAA 2004, Kingston, Canada, July 22-24, 2004, Revised Selected Papers*, pages 156–166, 2004. doi:10.1007/978-3-540-30500-2_15.

[JR93] Tao Jiang and Bala Ravikumar. Minimal NFA problems are hard. *SIAM J. Comput.*, 22(6):1117–1141, 1993. doi:10.1137/0222067.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, pages 85–103, 1972. doi:10.1007/978-1-4684-2001-2_9.

[KSV25] R. Keerthan, B. Srivathsan, and R. Venkatesh. An automaton model to succinctly represent suffix-based specifications of a concurrent system. In Salem Lahlou and Madhavan Mukund,

editors, *Networked Systems - 13th International Conference, NETYS 2025, Rabat, Morocco, May 21-23, 2025, Proceedings*, Lecture Notes in Computer Science, pages 129–145. Springer, 2025. doi:10.1007/978-3-032-00347-8_8.

[KSVV24] R. Keerthan, B. Srivathsan, R. Venkatesh, and Sagar Verma. Deterministic suffix-reading automata. In Antonis Achilleos and Adrian Francalanza, editors, *Proceedings Fifteenth International Symposium on Games, Automata, Logics, and Formal Verification, GandALF 2024, Reykjavik, Iceland, 19-21 June 2024*, volume 409 of *EPTCS*, pages 70–87, 2024. doi:10.4204/EPTCS.409.9.

[LMS22] Sylvain Lombardy, Victor Marsault, and Jacques Sakarovitch. *Awali, a library for weighted automata and transducers (version 2.2)*, 2022. Software available at http://vaucanson-project.org/Awali/2.2/.

[MMW09] Mehryar Mohri, Pedro J. Moreno, and Eugene Weinstein. General suffix automaton construction algorithm and space bounds. *Theor. Comput. Sci.*, 410(37):3553–3562, 2009. doi:10.1016/j.tcs.2009.03.034.

[VHL+12] Margus Veanes, Pieter Hooimeijer, Benjamin Livshits, David Molnar, and Nikolaj S. Bjørner. Symbolic finite state transducers: algorithms and applications. In *Proceedings of the 39th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2012, Philadelphia, Pennsylvania, USA, January 22-28, 2012*, pages 137–150, 2012. doi:10.1145/2103656.2103674.

[VSKA14] R. Venkatesh, Ulka Shrotri, G. Murali Krishna, and Supriya Agrawal. EDT: A specification notation for reactive systems. In *Design, Automation & Test in Europe Conference & Exhibition, DATE 2014, Dresden, Germany, March 24-28, 2014*, pages 1–6, 2014.

[ZLL02] Marc K. Zimmerman, Kristina Lundqvist, and Nancy G. Leveson. Investigating the readability of state-based formal requirements specification languages. In *Proceedings of the 24th International Conference on Software Engineering, ICSE 2002, 19-25 May 2002, Orlando, Florida, USA*, pages 33–43, 2002.