Deterministic Suffix-reading Automata

R Keerthan
Tata Consultancy Services Innovation Labs
Pune, India
Chennai Mathematical Institute, India
keerthan.r@tcs.com

B Srivathsan
Chennai Mathematical Institute, India
CNRS IRL 2000, ReLaX, Chennai, India
sri@cmi.ac.in

R Venkatesh
Sagar Verma
Tata Consultancy Services Innovation Labs
Pune, India
r.venky@tcs.com
verma.sagar2@tcs.com *

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. 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. We make a surprising observation 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.

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 [17], model-checking [4], software verification [2, 1, 9], and formal specification languages [12]. A central challenge in the application of automata is the size of the automaton involved. Non-determinism gives exponential succinctness, however, a deterministic model is useful in formal specifications and automata implementations. The literature offers different ways to get succinct representations of DFAs. We recall a few of them below and propose a new solution to this problem.

One of the reasons for large DFAs is 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 [18, 6] 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. Symbolic automata have been implemented in many tools and have been widely applied (see [5] for a list of tools and applications).

*All authors have contributed equally and are listed in the alphabetical order of last names.

To appear in EPTCS.

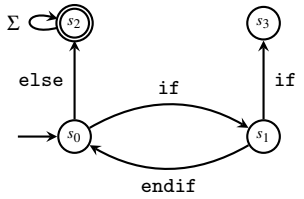

Figure 1: DSA for out-of-context `else`

Another dimension in reducing the DFA representation is to consider transitions on 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 [7], and later Hashiguchi [13] proved that for every regular language L there is a minimal GA in which the edge labels are at most a polynomial function in m, where m is size of the syntactic monoid of L. Giammarresi *et al.* [8] considers deterministic generalized automata (DGA) and proposes an algorithm to generate a minimal DGA (in terms of the number of states) in which the edges have length at most the size of the minimal DFA. The algorithm uses a method to suppress states and create longer labels. The key observation is that minimal DGAs can be derived from the canonical DFA by suppressing states.

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 a 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.

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 DSAs 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, number of states is not a faithful measure of the size of a DSA. As described in [8], 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:

1. Presentation of a definition of a new kind of automaton - DSA (Section 3).

2. Proof 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 1, Theorem 1). 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 minimal DFA has size quadratic in n, whereas size of DSAs is a linear function of n (Lemma 2).
3. We present a method to derive DSAs out of DFAs, a DFA-to-DSA conversion (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. Our main technical contribution lies in identifying sufficient conditions on the selected subset of states, so that the derivation procedure preserves the language (Theorem 9).
4. 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 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, 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).
5. Finally, we show that given a DFA and a number k, deciding if there exists a DSA of size $\leq k$ is NP-complete (Section 7).

Related work. The closest to our work is [8] 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 ideas presented in Section 6 of our work, also apply for state-minimality: the same example shows that in order to get 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 [8] 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 [8], and continues to remain so, to the best of our knowledge. Expression automata [11] allow regular expressions as transition labels. This model was already considered in [3] 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 [11] with restrictions that limit the expressive power. An algorithm to convert a DFA to a DEA, by repeated state elimination, is proposed in [11]. The resulting DEA is minimal in the number of states. The issue with Expression automata is the high expressivity of the transition condition, that makes states almost irrelevant. On the other hand, DEA have restrictions that make the model less expressive than DFAs. Minimization of NFAs was studied in [15] 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 [10].

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

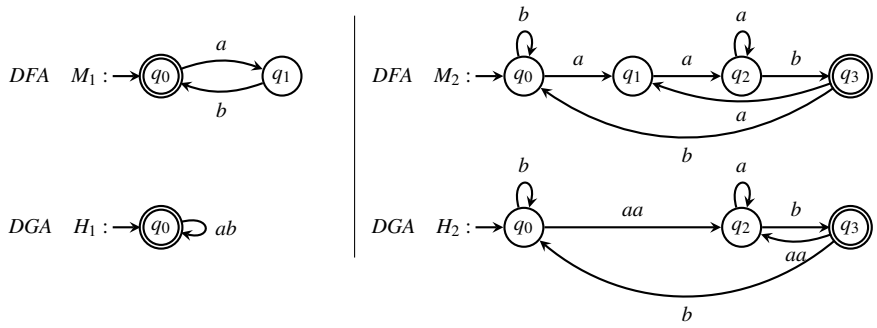

Figure 2: Examples of DFAs and corresponding DGAs, over alphabet $\{a,b\}$.

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 *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 $w = a_1 a_2 \dots a_n$ (where $a_i \in \Sigma$) is a sequence of transitions $(q_0, a_1, q_1)(q_1, a_2, q_2) \dots (q_{n-1}, a_n, q_n)$ where each $(q_i, a_{i+1}, q_{i+1}) \in \delta$ for $0 \leq i < n$, 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 a has unique run, or it has no run. The language $\mathscr{L}(M)$ 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 $\mathscr{L}(M)$. 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 [14] 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)* [8] 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 a atmost one run on every word. The language $\mathscr{L}(H)$ is the set of words with an accepting run. Figure 2 gives examples of DFAs and corresponding DGAs.

It was shown in [8] 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 $\mathscr{S}(H, S)$ 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.

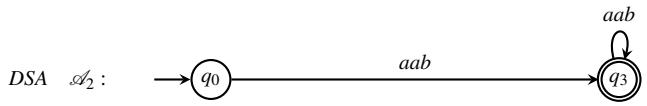

Figure 3: DSA $\mathscr{A}_2$ accepts $L_2 = \Sigma^* aab$, with $\Sigma = \{a, b\}$.

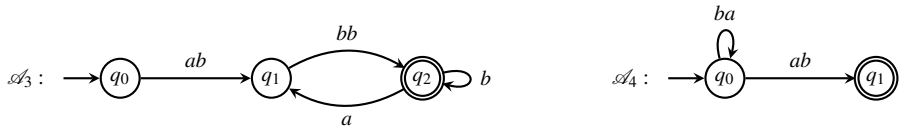

Figure 4: $\mathscr{A}_3$ accepts $L_3 = \Sigma^* ab \Sigma^* bb$ and $\mathscr{A}_4$ accepts $L_4 = (b^* ba)^* a^* ab$.

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 $\mathscr{A}_2$ 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 $\mathscr{A}_2$) to accept L_3 would look like $\mathscr{A}_3$ 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 1 (DSA)**.** *A* deterministic suffix-reading automaton (DSA) $\mathscr{A}$ *is 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 $|\mathscr{A}|$ *of DSA* $\mathscr{A}$ *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 $\mathscr{A}_4$ 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 2. *A* move *of DSA $\mathscr{A}$ is a pair (t,w) where $t=(q,\alpha,q')\in\Delta$ is a transition of $\mathscr{A}$ 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 contains a label in* $\mathrm{Out}(q)$ *as suffix.*

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. 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.

Definition 3. *A run of $\mathscr{A}$ 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 $q=q_0 \xrightarrow[\alpha_0]{w_0} q_1 \xrightarrow[\alpha_1]{w_1} \cdots \xrightarrow[\alpha_{m-1}]{w_{m-1}} q_m \xrightarrow{w_m}$ such that $w=w_0w_1\ldots w_{m-1}w_m$, 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 $\mathscr{L}(\mathscr{A})$ of $\mathscr{A}$ is the set of all words that have an accepting run starting from the initial state q^{init}.*

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 (Tracking DFA for a DSA.)**.** *For a DSA $\mathscr{A}=(Q^{\mathscr{A}},\Sigma,q_{in}^{\mathscr{A}},\Delta^{\mathscr{A}},F^{\mathscr{A}})$, we give a DFA $M_{\mathscr{A}}$, called its* tracking DFA. *For $q\in Q^{\mathscr{A}}$, let $\overline{\mathrm{Out}}(q)$ be the set of all prefixes of words in* $\mathrm{Out}(q)$. *States of $M_{\mathscr{A}}$ are given by: $Q^M=\bigcup_{q\in Q^{\mathscr{A}}}\{(q,\beta)\mid\beta\in\overline{\mathrm{Out}}(q)\}\cup q_{copy}$.*

The initial state is $(q_{in}^{\mathscr{A}},\varepsilon)$ and final states are $\{(q,\varepsilon)\mid q\in F^{\mathscr{A}}\}$. Transitions are as below: For every $q\in Q^{\mathscr{A}},\beta\in\overline{\mathrm{Out}}(q),a\in\Sigma$, 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 $(q,\beta',q')\in\Delta^{\mathscr{A}}$, ($q'$ may equal q also)*
- *$(q,\beta)\xrightarrow{a}(q,\beta')$ if $\beta'\notin\mathrm{Out}(q)$ and $\beta'\neq\varepsilon$,*
- *$(q,\beta)\xrightarrow{a}q_{copy}$ if $\beta'=\varepsilon$,*
- *$q_{copy}\xrightarrow{a}s$, 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).

Lemma 1. *For every DSA $\mathscr{A}$, the language $\mathscr{L}(\mathscr{A})$ equals the language $\mathscr{L}(M_{\mathscr{A}})$ of its tracking DFA.*

Lemma 1 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. We start with a family of languages for which DSAs are concise.

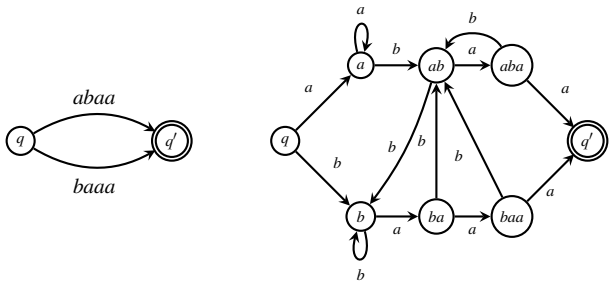

Figure 5: A DSA on the left, and the corresponding DFA for matching the strings *abaa* and *baaa*.

Lemma 2. *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 $4+2n$. Any DFA for L_n has size at least n^2.*

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 $2k(1+2|\Sigma|)$.

Theorem 1. *For a regular language L, let $n_F^{cmp}, n_F^{trim}, n_G^{trim}, n_S$ 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. We have:*

1. $\dfrac{n_F^{cmp}}{2(1+2|\Sigma|)} \leq n_S \leq n_F^{cmp}$
2. *no relation between n_S and n_F^{trim}, n_G^{trim}: there is a language for which n_S is the smallest, and another language for which n_S is the largest of the three.*

5 Suffix-tracking sets – obtaining DSA from DFA

For DGAs, a method to derive smaller DGAs by suppressing states was recalled in Section 2. Our goal is to investigate a similar procedure for DSAs. 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 [8], 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 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 useless transitions.

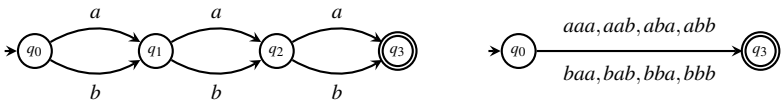

Figure 6: Suppressing states can add exponentially many labels and increase total size.

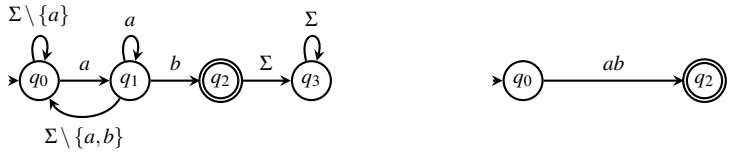

Figure 7: Suppressing states can sometimes reduce total size

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). 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 (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 from p to q modulo S*, as the set of all words $a_1 a_2 \ldots a_n \in \Sigma^+$ 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: $\{p_1, \ldots, p_{n-1}\} \subseteq Q \setminus S$, 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 $\mathsf{SP}(q_0 \rightsquigarrow q_2, S) = aa$.

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 6 (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 7 (Suffix-compatible transitions)**.** *Fix a subset $S \subseteq Q$. A transition $q \xrightarrow{a} u$ is suffix-compatible w.r.t. S if either of $q, u \in S$ **OR** $\forall p \in S$, and for every $\sigma \in \mathsf{SP}(p \rightsquigarrow q, S)$, there is an $\alpha \in \mathsf{SP}(p \rightsquigarrow u, S)$ s.t.:*

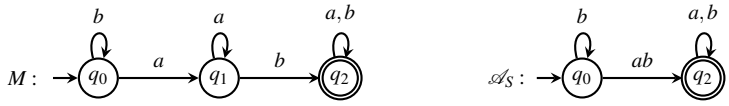

Figure 8: DFA M and an equivalent DSA $\mathscr{A}_S$ 'induced' with $S = \{q_0, q_2\}$.

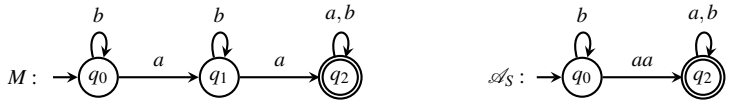

Figure 9: DFA M and DSA $\mathscr{A}_S$ 'induced' with $S = \{q_0, q_2\}$. Not equivalent.

- 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 σa, 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 $\mathrm{Out}(p,S)$.

Lemma 3. *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 contains any word from* $\mathrm{Out}(p,S)$ *as suffix, and*
- *there is* $\alpha \in \mathsf{SP}(p \rightsquigarrow p_n, S)$ *such that* α *is the longest suffix of w among words in* $\mathsf{SP}(p,S)$.

Lemma 4. *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 contains a word in* $\mathrm{Out}(p,S)$ *as suffix. 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).*

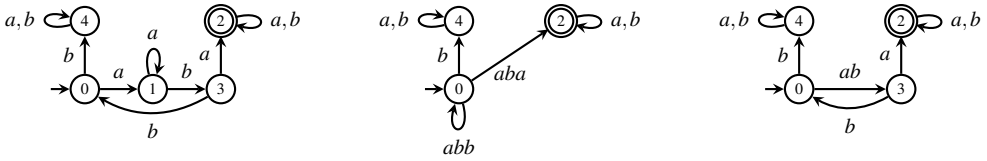

Figure 10: A DFA, a non-equivalent DSA and an equivalent induced DSA.

- *the longest suffix of w, among* $\mathsf{SP}(p,S)$ *lies in* $\mathsf{SP}(p \rightsquigarrow p_n,S)$.

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 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 the 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.

Definition 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 2. *Let S be a suffix-tracking set of complete DFA M, and let $\mathscr{A}_S$ be the DSA induced using S. Then: $\mathscr{L}(\mathscr{A}_S) = \mathscr{L}(M)$*

Proof. Pick $w \in \mathscr{L}(M)$. 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 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 3, and by the definition of induced DSA 6, no transition of $\mathscr{A}_S$ 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 $\mathscr{A}_S$. This shows there is a move $q_0 \xrightarrow[\alpha]{w_1 \dots w_i} q_i$ in $\mathscr{A}_S$. 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 $\mathscr{A}_S$ on the rest of the word. This shows $w \in \mathscr{L}(\mathscr{A}_S)$.

Pick $w \in \mathscr{L}(\mathscr{A}_S)$. There is an accepting run ρ of $\mathscr{A}_S$ starting at the initial state q_0. Consider the first move $q_0 \xrightarrow[\alpha]{w_1 \dots w_i} q_i$ of $\mathscr{A}_S$ on the word. By the semantics of a move (Definition 2) and Lemma 4, 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 $\mathscr{A}_S$ to get an accepting run of M. □

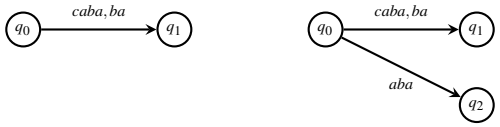

Figure 11: Illustrating bigger-suffix transitions and when they are useless

Removing some useless 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 $\mathscr{A}_S$ 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 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 useless 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 10. *Let $\mathscr{A}$ be a DSA, q, q' be states of $\mathscr{A}$ 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* useless *if it is not useful.*

We will say that t is a useless 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 useless. 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 useless, but the loop on q_0 in Figure 4 is useful. Lemmas 5 and 6 prove correctness of removing useless transitions.

Lemma 5. *Let $\mathscr{A}$ be a DSA, and let $t := q \xrightarrow{\alpha} q'$ be a useless bigger-suffix-transition. Let $\mathscr{A}'$ be the DSA obtained by removing t from $\mathscr{A}$. Then, $L(\mathscr{A}) = L(\mathscr{A}')$.*

Proof. To show $L(\mathscr{A}) \subseteq L(\mathscr{A}')$. Let $w \in L(\mathscr{A})$ 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 $\mathscr{A}'$ 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 useless bigger-suffix transition. Therefore, every move using (q, α, q') in $\mathscr{A}$ will now be replaced by (q, β, q') in $\mathscr{A}'$. Hence we get an accepting run in $\mathscr{A}'$, implying $w \in L(\mathscr{A}')$.

To show $L(\mathscr{A}') \subseteq L(\mathscr{A})$. Consider $w \in L(\mathscr{A}')$ 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 $\mathscr{A}'$. Notice that if $q \xrightarrow[\beta]{w_j} q'$ is a move in $\mathscr{A}'$, the same is a move in $\mathscr{A}$ when $\alpha \not\sqsupseteq_{\mathsf{sf}} w_j$. When $\alpha \sqsupseteq_{\mathsf{sf}} w_j$, 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 useless self-loops, we assume that the DFA that we obtain is well-formed (Definition 12) and has no useless 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 useless bigger-suffix-transitions, and then remove the useless self-loops.

Lemma 6. *Let $\mathscr{A}$ 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 $\mathscr{A}'$ obtained by removing t from $\mathscr{A}$ satisfies $\mathscr{L}(\mathscr{A}) = \mathscr{L}(\mathscr{A}')$.*

Proof. To show $\mathscr{L}(\mathscr{A}) \subseteq \mathscr{L}(\mathscr{A}')$. Let $w \in \mathscr{L}(\mathscr{A})$ 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 $\mathscr{A}'$. Hence $w \in L(\mathscr{A}')$.

To show $L(\mathscr{A}') \subseteq L(\mathscr{A})$. Let $w \in L(\mathscr{A}')$ and $\rho' := q_0 \xrightarrow{w_0} q_1 \xrightarrow{w_1} \cdots \xrightarrow{w_{m-1}} q_m$ be an accepting run in $\mathscr{A}'$. 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 $\mathscr{A}$. 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 $\mathscr{A}$ 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 $\mathscr{A}$. □

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 11 (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 useless transitions removed.*

By Theorem 2 and Lemma 5, we get the following result.

Theorem 3. *Every DSA that is derived from a complete DFA is language equivalent to it.*

Figure 12: Minimal DSA is not unique

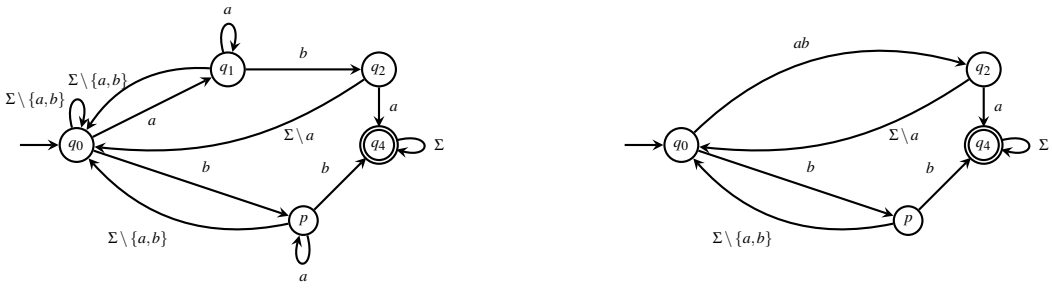

Figure 13: DFA M^* on the left and a derived DSA $\mathscr{A}^*_S$ with $S = \{q_0, q_2, q_4, p\}$ on the right.

6 Minimality, some observations and some challenges

Theorem 1 shows that we can not expect DSAs to be smaller than (trim) DFAs or DGAs in general. However, Lemma 2 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. The next simple observation is that a minimal DSA will not have useless transitions since removing them gives an equivalent DSA with strictly smaller size. In fact, we can assume a certain well-formedness condition on the minimal DSAs, in the same spirit as the definition of well-formed sets in our derivation procedure: 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.

Definition 12 (Well-formed DSA)**.** *A DSA $\mathscr{A}$ 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 $\beta \in \mathrm{Out}(q)$.*

Any transition violating well-formedness can be removed, without changing the language. Therefore, we can safely assume that minimal DSAs are well-formed. Due to the "well-formedness" property in suffix-tracking sets, the DSAs induced by suffix-tracking sets are naturally well-formed. Since removing useless transitions preserves this property, the DSAs that are derived using our DFA-to-DSA procedure (Definition 11) are well-formed. The next proposition shows that every DSA that is well-formed and has no useless transitions (and in particular, the minimal DSAs) can be derived from the corresponding tracking DFAs.

Proposition 1. *Every well-formed DSA with no useless transitions can be derived from its tracking DFA.*

Proposition 1 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,

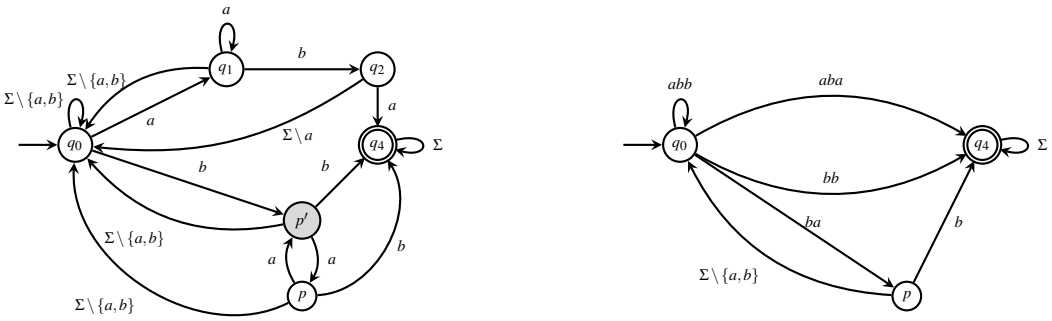

Figure 14: DFA M^{**} on the left and a derived DSA $\mathscr{A}_S^{**}$ with $S = \{q_0, q_2, q_4, p\}$ on the right.

then the set is not well-formed (see Definition 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 $\mathscr{A}_S^*$. We deduce $\mathscr{A}_S^*$ 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 $\mathscr{A}_S^{**}$ is shown in the right of the figure. The "heavy" transition on $\Sigma \setminus a$ disappears. There are some extra transition, like $q_0 \xrightarrow{bb} q_4$, but if Σ is large enough, the size of $\mathscr{A}_S^{**}$ will be smaller than $\mathscr{A}_S^*$. 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 Complexity of minimization

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

Theorem 4. *Given a DFA M and positive integer k, deciding whether there exists an equivalent DSA of total size $\leq k$ 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 $\mathscr{O}(k \cdot |\Sigma|)$ 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 [16]. 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 $k = (k'+2) \times 2\Delta + (2\Delta - 1)$. Here, Δ is a sufficiently large polynomial in $|V|, |E|$ which we will explain later.

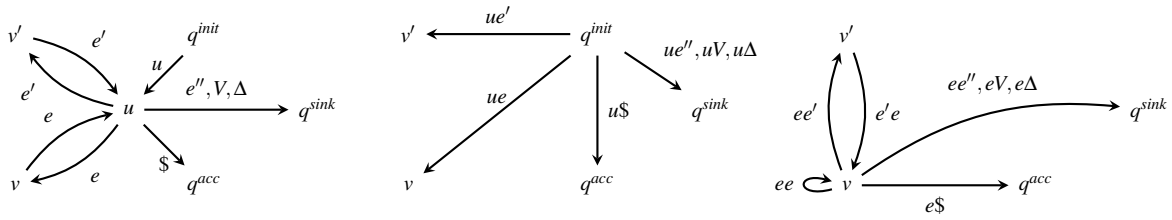

Figure 15: Left: Illustration of the neighbourhood of state u in the DFA M_G. Middle, Right: Transitions induced from q^{init} and v, on removing u.

The alphabet Σ_G is given by $V \cup E \cup \{\$\} \cup D$ where $D = \{1, 2, \ldots, \Delta\}$. 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 15 (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 15, we show the induced transitions from q^{init} and a vertex v. However, some of them will be useless transitions: most importantly, the set of transitions $q^{init} \xrightarrow{u1,u2,\ldots,u\Delta} q^{sink}$ will be useless bigger-suffix-transitions due to $q^{init} \xrightarrow{1,2,\ldots,\Delta} q^{sink}$. Similarly, $v \xrightarrow{e1,e2,\ldots,e\Delta} q^{sink}$ will be removed. There are some more useless 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 useless transitions, the Δ transitions from u no longer appear. If we choose Δ 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 Δ, DSAs with smallest (total) size are indeed obtained from suffix-tracking sets with the least number of states. Let $k = (k' + 2) \times 2\Delta + (2\Delta - 1)$.

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 $q \xrightarrow{1,2,\ldots,\Delta} q^{sink}$ from each except for the q^{sink} state. The transitions on q^{sink} are removable, and hence will be absent. All of this adds $(p+2) \times 2\Delta$ 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 Δ (say $(|V|+|E|)^4$), so that the size of these extra transitions is at most $2\Delta - 1$. Hence, total size is $\leq (p+2) \times 2\Delta + (2\Delta - 1)$.

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

DSA $\leq k$ implies vertex cover $\leq k'$. Let $\mathscr{A}$ be a DSA with size $\leq k$. It may not be derived from M_G. However, by Proposition 1 we know $\mathscr{A}$ is derived from a DFA M, the tracking DFA for $\mathscr{A}$. 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 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 2Δ in the total size, due to the Δ transitions. We will also have these Δ transitions from the initial and accepting states. Therefore, the total size is $(p+2) \times 2\Delta + y$ for some $y > 0$. Hence $(p+2) \times 2\Delta \leq k$. This implies $p \leq k'$: otherwise we will have $p \geq k'+1$, and hence $(p+2) \times 2\Delta \geq (k'+1+2) \times 2\Delta = (k'+2) \times 2\Delta + 2\Delta > k$, a contradiction.

8 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 [8] 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. This leads to several questions about the DSA model, and our derivation methodology.

When does the smallest DSA derived from the canonical DFA correspond to a minimal DSA? Can we use our techniques to study minimality in terms of number of states? Closure properties of DSAs - do we perform the union, intersection and complementation operations on DSAs without computing the entire equivalent DFAs? What about practical studies of using DSAs? 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

[1] Ahmed Bouajjani, Peter Habermehl & Tomás Vojnar (2004): *Abstract Regular Model Checking*. In: *Computer Aided Verification, 16th International Conference, CAV 2004, Boston, MA, USA, July 13-17, 2004, Proceedings*, pp. 372–386, doi:10.1007/978-3-540-27813-9_29.

[2] Ahmed Bouajjani, Bengt Jonsson, Marcus Nilsson & Tayssir Touili (2000): *Regular Model Checking*. In: *Computer Aided Verification, 12th International Conference, CAV 2000, Chicago, IL, USA, July 15-19, 2000, Proceedings*, pp. 403–418, doi:10.1007/10722167_31.

[3] Janusz A. Brzozowski & Edward J. McCluskey (1963): *Signal Flow Graph Techniques for Sequential Circuit State Diagrams*. *IEEE Trans. Electron. Comput.* 12(2), pp. 67–76, doi:10.1109/PGEC.1963.263416.

[4] Edmund M. Clarke, Orna Grumberg, Daniel Kroening, Doron A. Peled & Helmut Veith (2018): *Model checking, 2nd Edition*. MIT Press. Available at `https://mitpress.mit.edu/books/model-checking-second-edition`.

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

[6] Loris D'Antoni & Margus Veanes (2017): *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*, pp. 47–67, doi:10.1007/978-3-319-63387-9_3.

[7] Samuel Eilenberg (1974): *Automata, languages, and machines. A*. Pure and applied mathematics, Academic Press. Available at `https://www.worldcat.org/oclc/310535248`.

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

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

[10] Noa Globerman & David Harel (1996): *Complexity Results for Two-Way and Multi-Pebble Automata and their Logics*. *Theor. Comput. Sci.* 169(2), pp. 161–184, doi:10.1016/S0304-3975(96)00119-3.

[11] Yo-Sub Han & Derick Wood (2004): *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*, pp. 156–166, doi:10.1007/978-3-540-30500-2_15.

[12] David Harel (1987): *Statecharts: A Visual Formalism for Complex Systems*. *Sci. Comput. Program.* 8(3), pp. 231–274, doi:10.1016/0167-6423(87)90035-9.

[13] Kosaburo Hashiguchi (1991): *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*, pp. 641–648, doi:10.1007/3-540-54233-7_170.

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

[15] Tao Jiang & Bala Ravikumar (1993): *Minimal NFA Problems are Hard*. *SIAM J. Comput.* 22(6), pp. 1117–1141, doi:10.1137/0222067.

[16] Richard M. Karp (1972): *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*, pp. 85–103, doi:10.1007/978-1-4684-2001-2_9.

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

[18] Margus Veanes, Pieter Hooimeijer, Benjamin Livshits, David Molnar & Nikolaj S. Bjørner (2012): *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, pp. 137–150, doi:10.1145/2103656.2103674.