Once-Marking and Always-Marking 1-Limited Automata

Giovanni Pighizzini
Dipartimento di Informatica
Università degli Studi di Milano, Italy
pighizzini@di.unimi.it

Luca Prigioniero
Department of Computer Science
Loughborough University, UK
l.prigioniero@lboro.ac.uk

Single-tape nondeterministic Turing machines that are allowed to replace the symbol in each tape cell only when it is scanned for the first time are also known as 1-limited automata. These devices characterize, exactly as finite automata, the class of regular languages. However, they can be extremely more succinct. Indeed, in the worst case the size gap from 1-limited automata to one-way deterministic finite automata is double exponential.

Here we introduce two restricted versions of 1-limited automata, *once-marking* 1*-limited automata* and *always-marking* 1*-limited automata*, and study their descriptional complexity. We prove that once-marking 1-limited automata still exhibit a double exponential size gap to one-way deterministic finite automata. However, their deterministic restriction is polynomially related in size to two-way deterministic finite automata, in contrast to deterministic 1-limited automata, whose equivalent two-way deterministic finite automata in the worst case are exponentially larger. For always-marking 1-limited automata, we prove that the size gap to one-way deterministic finite automata is only a single exponential. The gap remains exponential even in the case the given machine is deterministic.

We obtain other size relationships between different variants of these machines and finite automata and we present some problems that deserve investigation.

1 Introduction

In 1967, with the aim of generalizing the concept of determinism for context-free languages, Hibbard introduced *limited automata*, a restricted version of Turing machines [3]. More precisely, for each fixed integer $d \geq 0$, a *d-limited automaton* is a single-tape nondeterministic Turing machine that is allowed to replace the content of each tape cell only in the first d visits.

Hibbard proved that, for each $d \geq 2$, d-limited automata characterize the class of context-free languages. For $d = 0$ these devices cannot modify the input tape, hence they are two-way finite automata, so characterizing regular languages. Furthermore, also 1-limited automata are no more powerful than finite automata. The proof of this fact can be found in [19, Thm. 12.1].

The investigation of these models has been reconsidered in the last decade, mainly from a descriptional point of view. Starting with [8, 9], several works investigating properties of limited automata and their relationships with other computational models appeared in the literature (for a recent survey see [7]).

In this paper we focus on 1-limited automata. We already mentioned that these devices are no more powerful than finite automata, namely they recognize the class of regular languages. However, they can be dramatically more succinct than finite automata. In fact, a double exponential size gap from 1-limited automata to one-way deterministic finite automata has been proved [8]. In other words, every n-state 1-limited automaton can be simulated by a one-way deterministic automaton with a number of states which is double exponential in n. Furthermore, in the worst case, this cost cannot be reduced.

Zs. Gazdag, Sz. Iván, G. Kovásznai (Eds.): 16th International Conference on Automata and Formal Languages (AFL 2023)
EPTCS 386, 2023, pp. 215–227, doi:10.4204/EPTCS.386.17

As pointed out in [8], this double exponential gap is related to a double role of the nondeterminism in 1-limited automata. When the head of a 1-limited automaton reaches for the first time a tape cell, it replaces the symbol in it according to a nondeterministic choice. Furthermore, the set of nondeterministic choices allowed during the next visits to the same cell depends on the symbol written in the first visit and that cannot be further changed, namely it depends on the nondeterministic choice made during the first visit.

With the aim of better understanding this phenomenon, we started to investigate some restrictions of 1-limited automata. On the one hand, we are interested in finding restrictions that reduce this double exponential gap to a single exponential. We already know that this happens for *deterministic* 1-limited automata [8]. So the problem is finding some restrictions that, still allowing nondeterministic transitions, avoid the double exponential gap. On the other hand, we are also interested in finding some very restricted forms of 1-limited automata for which a double exponential size gap in the conversion to one-way deterministic automata remains necessary in the worst case.

A first attempt could be requiring deterministic rewritings, according to the current configuration of the machine, every time cells are visited for the first time, still keeping nondeterministic the choice of the next state and head movement. Another attempt could be to allow nondeterministic choices for the symbol to rewrite, but not for the next state and the head movement. In both cases the double exponential gap to one-way deterministic finite automata remains possible. Indeed, in both cases, different computation paths can replace the same input prefix on the tape with different strings, as in the original model. Actually, we noticed that the double exponential gap can be achieved already for 1-limited automata that, in each computation, have the possibility to mark just one tape cell leaving the rest of the tape unchanged. This inspired us to investigate machines with such a restriction, which we call *once-marking* 1*-limited automata*. We show that the double exponential size gap to one-way deterministic finite automata remains possible even for once-marking 1-limited automata that are *sweeping* (namely, change the head direction only at the left or right end of the tape) and that are allowed to use nondeterminism only in the first visit to tape cells. Comparing the size of once-marking 1-limited automata with other kinds of finite automata, we prove an exponential gap to two-way nondeterministic automata. The situation changes significantly when nondeterministic transitions are not possible. Indeed, we prove that every deterministic once-marking 1-limited automaton can be converted into an equivalent two-way deterministic finite automaton with only a polynomial size increasing. The costs we obtain concerning once-marking 1-limited automata are summarized in Figure 2.

As mentioned above, the double exponential gap from 1-limited automata to one-way deterministic finite automata is related to the fact that different computation paths can replace the same input prefix on the tape with different strings. This suggested the idea of considering a different restriction, which prevents this possibility, by requiring the replacement of each input symbol a with a symbol that depends only on a. To this aim, here we introduce *always-marking* 1*-limited automata*, that in the first visit replace each symbol with a marked version of it. We show that in this case the gap from these devices, in the nondeterministic version, to one-way deterministic finite automata reduces to a single exponential. The same gap holds when converting always-marking 1-limited automata into one-way nondeterministic finite automata, but even when converting *deterministic* always-marking 1-limited automata into *two-way nondeterministic* finite automata. The bounds we obtain concerning always-marking 1-limited automata are summarized in Figure 3.

The paper is organized as follows. After presenting in Section 2 the preliminary notions used in the paper and, in particular, the definition of 1-limited automata with the fundamental results on their descriptional complexity, in Section 3 we introduce once-marking and always-marking 1-limited automata,

together with some witness languages that will be useful to obtain our results. Sections 4 and 5 are devoted to the investigation of the descriptional complexity of these models. We conclude the paper presenting some final remarks and possible lines for future investigations.

2 Preliminaries

In this section we recall some basic definitions useful in the paper. Given a set S, $\#S$ denotes its cardinality and 2^S the family of all its subsets. Given an alphabet Σ, a string $w \in \Sigma^*$, and a symbol $a \in \Sigma$, $|w|$ denotes the length of w, Σ^k the set of all strings on Σ of length k, $\dot{a}$ the *marked versions* of a, and $\dot{\Sigma} = \{\dot{a} \mid a \in \Sigma\}$ the set of the marked versions of the symbols in Σ.

We assume the reader familiar with notions from formal languages and automata theory, in particular with the fundamental variants of finite automata (1DFAs, 1NFAs, 2DFAs, 2NFAs, for short, where 1/2 mean *one-way/two-way* and D/N mean *deterministic/nondeterministic*, respectively). For any unfamiliar terminology see, e.g., [4].

A 1-*limited automaton* (1-LA, for short) is a tuple $\mathscr{A} = (Q, \Sigma, \Gamma, \delta, q_I, F)$, where Q is a finite *set of states*, Σ is a finite *input alphabet*, Γ is a finite *work alphabet* such that $\Sigma \cup \{\rhd, \lhd\} \subseteq \Gamma$, $\rhd, \lhd \notin \Sigma$ are two special symbols, called the *left* and the *right end-markers*, and $\delta : Q \times \Gamma \to 2^{Q \times (\Gamma \setminus \{\rhd, \lhd\}) \times \{-1, +1\}}$ is the *transition function*. At the beginning of the computation, the input word $w \in \Sigma^*$ is stored onto the tape surrounded by the two end-markers, the left end-marker being in position zero and the right end-marker being in position $|w| + 1$. The head of the automaton is on cell 1 and the state of the finite control is the *initial state* q_I.

In one move, according to δ and the current state, $\mathscr{A}$ reads a symbol from the tape, changes its state, replaces the symbol just read from the tape by a new symbol, and moves its head to one position forward or backward. Furthermore, the head cannot pass the end-markers, except at the end of computation, to accept the input, as explained below. Replacing symbols is allowed to modify the content of each cell only during the first visit, with the exception of the cells containing the end-markers, which are never modified. Hence, after the first visit, a tape cell is "frozen".[1]

The automaton $\mathscr{A}$ accepts an input w if and only if there is a computation path that starts from the initial state q_I with the input tape containing w surrounded by the two end-markers and the head on the first input cell, and that ends in a *final state* $q \in F$ after passing the right end-marker. The device $\mathscr{A}$ is said to be *deterministic* (D-1-LA, for short) whenever $\#\delta(q, \sigma) \leq 1$, for any $q \in Q$ and $\sigma \in \Gamma$.

Two-way finite automata are limited automata in which no rewritings are possible. On the other hand, one-way finite automata can scan the input in a one-way fashion only. A finite automaton is, as usual, a tuple $(Q, \Sigma, \delta, q_I, F)$, where, analogously to 1-LAs, Q is the finite set of states, Σ is the finite input alphabet, δ is the transition function, q_I is the initial state, and F is the set of final states. We point out that for two-way finite automata we assume the same accepting conditions as for 1-LAs.

Two-way machines in which the direction of the head can change only at the end-markers are said to be *sweeping* [18].

In this paper we are interested to compare the size of machines. The *size* of a model is given by the total number of symbols used to write down its description. Therefore, the size of 1-LAs is bounded by

[1] More technical details can be found in [8]. However, a syntactical restriction forcing 1-LAs to replace in the first visit to each tape cell the input symbol in it with another symbol from an alphabet Γ_1 disjoint from Σ, was given. Here we drop this restriction, in order to be able to see once-marking 1-LAs as a restriction of 1-LAs. It is always possible to transform a 1-LA into an equivalent 1-LA satisfying such a syntactical restriction, just extending Γ with a marked copy of Σ and suitably modifying the transition function.

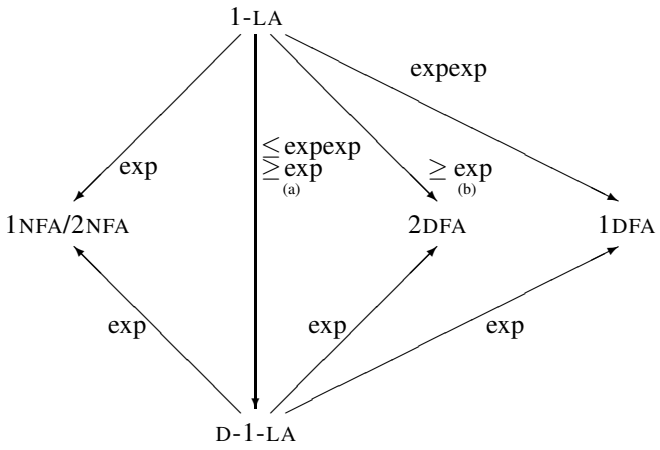

Figure 1: Size costs of conversions of 1-LAs and D-1-LAs into equivalent one-way and two-way deterministic and nondeterministic finite automata. For all the costs upper and matching lower bounds have been proved, with the only exception of (a) and (b), for which the best known lower and upper bounds are, respectively, exponential and double exponential.

a polynomial in the number of states and of work symbols, while, in the case of finite automata, since no writings are allowed, the size is linear in the number of instructions and states, which is bounded by a polynomial in the number of states and in the number of input symbols.

The size costs of the simulations from 1-LAs to finite automata have been studied in [8] and are summarized in Figure 1.

3 Witness Languages and Variants of 1-Limited Automata

As mentioned in the introduction, 1-LAs can be very succinct. In fact, for some languages the size gap to 1DFA is double exponential. We already observed that this gap is related to nondeterminism. Indeed, if nondeterministic choices are not possible, the gap reduces to a single exponential (see Figure 1). However, we want to understand better on the one hand how much we can restrict the model, still keeping this double exponential gap and, on the other hand, if there is a restriction that, still allowing some kind of nondeterminism, reduces the gap to a single exponential.

In our investigations, the following language, which is defined with respect to an integer parameter $n > 0$, will be useful:

$$K_n = \{x_1 \cdots x_k \cdot x \mid k > 0,\ x_1, \ldots, x_k, x \in \{a,b\}^n,\ \exists j \in \{1, \ldots, k\},\ x_j = x\}.$$

We point out that each string in the language is a list of blocks of length n. We ask the membership of the last block to the list of previous ones.

Theorem 1. *The language K_n is accepted by a* 1-LA *with $O(n)$ states that, in each accepting computation, replaces the content only of one cell.*

Proof. A 1-LA $\mathscr{M}$ can scan the tape from left to right, marking a nondeterministically chosen tape cell. In this scan, $\mathscr{M}$ can also verify that the input length is a multiple of n. Furthermore, the marking can be done in the last cell of a block of length n. For this phase $O(n)$ states are enough.

Then the machine has to compare the symbols in the last block with the symbols in the chosen one, namely the block which ends with the marked cell. This can be done by moving the head back and forth from the last block to the chosen block, comparing the symbols in the corresponding positions in the two blocks, and rejecting in case of mismatch. Again, this can be implemented, using a counter modulo n, with $O(n)$ states. □

Using standard distinguishability arguments, it can be proved that to accept K_n, a 1DFA requires a number of states double exponential in n (state lower bounds for K_n are summarized in Theorem 2 below).

Hence, the language K_n is a witness of the double exponential gap from 1-LAs to 1DFAs. From Theorem 1, we can notice that this gap is obtained by using the capabilities of 1-LAs in a very restricted way: during each accepting computation, only the content of one cell is modified. This suggested us to considering the following restricted version of 1-LAs:

Definition 1. *A* 1-LA *is said to be* once marking *if in each computation there is a unique tape cell whose input symbol σ is replaced with its marked version $\dot{\sigma}$, while all the remaining cells are never changed.*

In the following, for brevity, we indicate once-marking 1-LAs and once-marking D-1-LAs as OM-1-LAs and D-OM-1-LAs, respectively.

We shall consider another restriction, in which the 1-LA marks, in the first visit, every cell reached by the head.

Definition 2. *A* 1-LA *is said to be* always marking *if, each time the head visits a tape cell for the first time, it replaces the input symbol σ in it with its marked version $\dot{\sigma}$.*

In the following, for brevity, we indicate always-marking 1-LAs and always-marking D-1-LAs as AM-1-LAs and D-AM-1-LAs, respectively.

We point out that OM-1-LAs and AM-1-LAs use the work alphabet $\Gamma = \Sigma \cup \dot{\Sigma} \cup \{\triangleright, \triangleleft\}$. Hence, the relevant parameter for evaluating the size of these devices is their number of states, differently than 1-LAs, in which the size of the work alphabet is not fixed.

We present another language that will be used in the paper. As K_n, it is defined with respect to a fixed integer $n > 0$:

$$J_n = \{x \cdot x_1 \cdots x_k \mid k > 0,\ x_1, \ldots, x_k, x \in \{a,b\}^n,\ \exists j \in \{1, \ldots, k\},\ x_j = x\}.$$

Even in this case, a string is a list of blocks of length n. Here we ask the membership of the first block to the subsequent list. Notice that J_n is the reversal of K_n.

We have the following lower bounds:

Theorem 2. *Let $n > 0$ be an integer.*

- *To accept J_n,* 1DFA*s and* 1NFA*s need at least 2^n states, while* 2NFA*s need at least $2^{\frac{n-1}{2}}$ states.*
- *To accept K_n,* 1DFA*s need 2^{2^n} states,* 1NFA*s need at least 2^n states, and* 2NFA*s need at least $2^{\frac{n-1}{2}}$ states.*

Proof. (sketch) The lower bounds for one-way machines can be proved using standard distinguishability arguments and the fooling set technique [1] (see [8, 13] for similar proofs with slightly different languages).

Using a standard conversion, from a k-state 2NFA accepting K_n we can obtain an equivalent 1DFA with no more than 2^{k+k^2} states [14, 16]. Since every 1DFA accepting K_n should have at least 2^{2^n} states, we

get that $k+k^2 \geq 2^n$. Hence k grows as an exponential in n. In particular, it can be verified that $k > 2^{\frac{n-1}{2}}$. Since from each 2NFA accepting a language we can easily obtain a 2NFA with a constant amount of extra states accepting the reversal of such a language, we can conclude that the number of states of each 2NFA accepting J_n or K_n must be at least exponential in n. □

4 Once-Marking 1-Limited Automata

During each computation, *once-marking 1-limited automata* are able to mark just one input cell.

From Theorem 1, we already know that the language K_n can be accepted by a OM-1-LA with $O(n)$ states. We now show that such a machine can be turned in a even more restricted form:

Theorem 3. *The language K_n is accepted by a* OM-1-LA *with $O(n)$ states that is sweeping and uses nondeterministic transitions only in the first traversal of the tape.*

Proof. We discuss how to modify the $O(n)$-state OM-1-LA $\mathscr{M}$ described in the proof of Theorem 1 in order obtain a sweeping machine that uses nondeterministic transitions only in the first sweep. $\mathscr{M}$ makes a first scan of the input, exactly as described in the proof of Theorem 1. In this scan the head direction is never changed. When the right end-marker is reached, $\mathscr{M}$ makes n iterations, which in the following description will be counted from 0 to $n-1$.

The purpose of the iteration i, $i = 0,\ldots,n-1$, is to compare the $(n-i)$th symbols of the last block and of the chosen one. To this aim, the iteration starts with the head on the right end-marker, and uses a counter modulo n, initialized to $(i+1) \bmod n$. The counter is decremented while moving to the left. In this way, it contains 0 exactly while visiting the $(n-i)$th cell of each input block. Hence, the automaton can easily locate the $(n-i)$th symbols of the last block and of the chosen one and check if they are equal. Once the left end-marker is reached, $\mathscr{M}$ can cross the tape from left to right, remembering the number i of the iteration. Notice that $\mathscr{M}$ does not need to keep this number while moving from right to left. Indeed the value of i can be recovered from the value of the counter when the left end-marker is reached.

Once the iteration i is completed, if the last check was unsuccessful then $\mathscr{M}$ can stop and reject. Otherwise it can start the next iteration, if $i < n-1$, or accepts.

From the discussion above, it can be easily verified that $\mathscr{M}$ is sweeping, makes nondeterministic choices only in the first sweep, and has $O(n)$ many states. □

We now study the size relationships between OM-1-LAs and finite automata. First, we observe that OM-1-LAs can be simulated by 1NFAs and by 1DFAs at the costs of an exponential and a double exponential increase in the number of states, respectively. These upper bounds derive from the costs of the simulations of 1-LAs by finite automata presented in [8, Thm. 2]. By considering the language K_n, we can conclude that these costs cannot be reduced:

Theorem 4. *Let $\mathscr{M}$ be a n-state* OM-1-LA*s. Then $\mathscr{M}$ can be simulated by a* 1NFA *and by a* 2NFA *with a number of states exponential in n, and by a* 1DFA *with a number of states double exponential in n. In the worst case these costs cannot be reduced.*

Proof. The upper bounds derive from the cost of the simulations of 1-LAs by 1NFAs and 1DFAs given in [8, Thm. 2]. For the lower bounds we consider the language K_n. As proved in Theorem 3, this language can be accepted by a OM-1-LA with $O(n)$ states. Furthermore, according to Theorem 2, it requires a number of state exponential in n to be accepted by 1NFAs or 2NFAs, and a number of states double exponential in n to be accepted by 1DFAs. □

From Theorem 4, it follows that the ability of marking only once can give already a huge descriptional power. Furthermore, from Theorem 3, we can observe that this power is achievable even with a sweeping machine that does not use nondeterminism after the first sweep. From the size costs of the simulation of 1-LAs by finite automata (see Figure 1), we already know that nondeterminism is essential to obtain this huge descriptional power. We now prove that, without nondeterminism, the descriptional power on OM-1-LAs dramatically reduces:

Theorem 5. *For each n-state* D-OM-1-LA *there exists an equivalent* 2DFA *with* $O\left(n^3\right)$ *states.*

Proof. Let $\mathscr{A} = (Q, \Sigma, \Gamma, \delta, q_I, F)$ be a n-state D-OM-1-LA. We give a construction of an equivalent 2DFA $\mathscr{A}'$. Before doing that, let us introduce, from an high-level perspective, how the simulating machine works.

The 2DFA $\mathscr{A}'$ operates in different modes.

In the first part of the computation, before $\mathscr{A}$ marks *one* cell, $\mathscr{A}'$ is in beforeMarking mode, in which it simulates directly each transition of $\mathscr{A}$.

When $\mathscr{A}'$ has to simulate the transition $\delta(s,\sigma) = (\dot{\sigma}, d)$ used by $\mathscr{A}$ for marking a cell, besides changing its state and moving its head according to the transition, $\mathscr{A}'$ switches to afterMarking mode and stores in its finite control the symbol σ that has been marked and the state s in which $\mathscr{A}$ was immediately before the marking.

While in afterMarking mode, every time a cell is visited, $\mathscr{A}'$ has to select which transition of $\mathscr{A}$ to simulate depending on the symbol a scanned by the input head. There are two possibilities: if the scanned symbol is different than the symbol σ that has been marked, then the transition is simulated directly. Otherwise, $\mathscr{A}'$ switches to backwardSimulation mode (described later) to verify whether the current cell is the one that has been marked by $\mathscr{A}$. If this is the case, then $\mathscr{A}'$ simulates the transition of $\mathscr{A}$ on the marked symbol $\dot{\sigma}$, otherwise it simulates the transition on σ. In both cases $\mathscr{A}'$ keeps working in afterMarking mode, so selecting transitions according to the strategy described above, until there are no more moves to simulate. Therefore $\mathscr{A}'$ accepts if the last simulated transition corresponds to a right transition passing the right end-marker while simulating a final state of $\mathscr{A}$.

We now give some details on the backwardSimulation mode, which is the core of the simulation. We remind the reader that $\mathscr{A}'$ switches to this mode when, being in afterMarking mode, the input head is on a cell containing the symbol σ, which has been saved at the end of the beforeMarking mode. Let us indicate by j the current position of the head, namely the position that has to be verified.

The 2DFA $\mathscr{A}'$ has to verify whether j is the cell that has been marked by $\mathscr{A}$. To make this check, $\mathscr{A}'$ can verify whether the computation path of $\mathscr{A}$ on the given input reaches, from the initial configuration, a configuration with state s and the head on the currently scanned cell j (we remind the reader that s and σ have been saved in the control of $\mathscr{A}'$ when switching from beforeMarking to afterMarking mode), whose position, however, cannot be saved in the control.

To be sure that the machine does not "loses track" of the position j while performing this search, we use the following strategy:

- $\mathscr{A}'$ simulates a backward computation from the state s and the current position j.
- If the initial configuration of $\mathscr{A}$ is reached, then the cell from which the check has started is the one where the marking transition has been executed.
- At that point, the position j is recovered by "rolling back" the backward computation. This is done by repeating the (forward) computation of $\mathscr{A}$ from the initial configuration until a marking transition is used. In fact, since $\mathscr{A}$ is deterministic and once marking, this transition is necessarily

the one that, from the state s, marked σ. In other words, the forward computation of $\mathscr{A}$ that is simulated here is the same simulated in beforeMarking mode.

As we shall explain later, even in the case the initial configuration of $\mathscr{A}$ is not reached (namely the verification is not successful), our technique allows to recover the head position j from which the backward simulation started,

It is important to observe two key points for which this approach works. The first one is that OM-1-LAs mark only one cell during their computation. The second observation is that the simulated machine is deterministic. Therefore, along every accepting computation path from the initial configuration, it occurs only once that the symbol σ is scanned while $\mathscr{A}$ is in state s, which is when $\mathscr{A}$ makes a marking transition.

To make such a verification, and in particular the backward search, we use a technique originally introduced by Sipser [17]. This simulation has been then refined by Geffert, Mereghetti, and Pighizzini, which proved that 2DFAs can be made halting with a linear increase of the number of states [2]. In the following, we shall refer to the latter simulation as the *original simulation* and use the notation and terminology contained in [2], to which we address the interested reader for missing details.

The main difference with the original simulation is that there the simulating machine starts from the final configuration of the simulated device, because the goal is to verify the presence of an accepting computation path. In our case, the machine $\mathscr{A}'$ starts the backward simulation from the state s and the cell containing σ that has to be checked.

In the following, a *configuration* is a pair (q,i), where q is the current state and i is the position of the tape head.

Consider the graph whose nodes represent configurations and edges computation steps. Since $\mathscr{A}$ is deterministic, the component of the graph containing (s,j) is a tree rooted at this configuration, with backward paths branching to all possible predecessors of (s,j). In addition, no backward path starting from (s,j) can loop (hence, it is of finite length), because the marking configuration (s,j) cannot be reached by a forward path from a loop (due to the fact that the machine is deterministic).

The simulating machine $\mathscr{A}'$ can perform a depth-first search of this tree in order to detect whether the initial configuration $(q_I,0)$ belongs to the predecessors of (s,j). If this is the case, then the machine returns to the position j, by performing a forward simulation of $\mathscr{A}$ from $(q_I,0)$ until when s is entered while reading the symbol σ. We stress that this approach works because the simulated machine is deterministic. After that, the simulation of $\mathscr{A}$ in afterMarking mode is recovered by performing a move on the symbol $\dot{\sigma}$. On the other hand, if the whole tree has been examined without reaching $(q_I,0)$, then the cell in position j is not the marked one, so the machine simulates a move of $\mathscr{A}'$ on σ from the cell in position j, again switching back to afterMarking mode. Notice that this case occurs when there are no more predecessors of (s,j) to visit. So, in this case, the machine $\mathscr{A}'$ completes the depth-first search on the cell in position j, while looking for further nodes of the graph reachable from the configuration (s,j). Hence, no extra steps are required to retrieve the position j.

In conclusion, $\mathscr{A}'$ has three state components of size $O(n)$: one used in beforeMarking and afterMarking for the direct simulation of the transitions of $\mathscr{A}$, one for storing the state s and the symbol σ, and one used in backwardSimulation mode. So, the total number of states of $\mathscr{A}'$ is $O\left(n^3\right)$. □

In Figure 2 the state costs of the conversions involving OM-1-LAs are summarized. In particular, we proved that the size gap from OM-1-LAs to 2NFAs is exponential and to 1DFAs is double exponential, while D-OM-1-LAs and 2DFAs are polynomially related in size.

Some questions remain open, in particular about the costs of the simulations of OM-1-LAs by D-OM-1-LAs and by 2DFAs. At the moment, from the above mentioned results, we can derive double

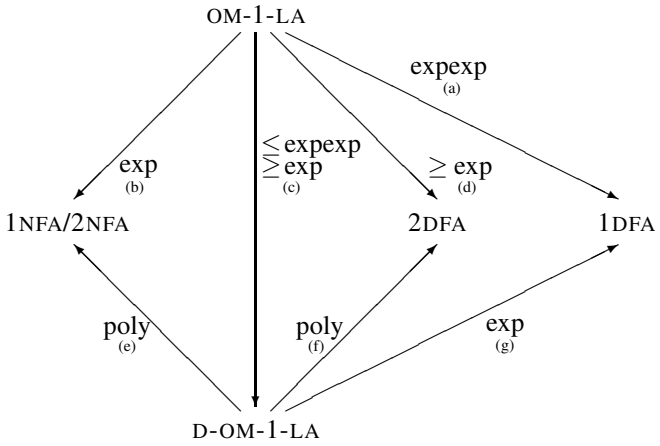

Figure 2: Size costs of conversions involving OM-1-LAs. The gaps (a) and (b) derive from Theorem 4. For (c) and (d) the lower bound derives from the lower bound of the language K_n on 2NFAs (Theorem 2); the best known upper bound derives from (a). The bounds (e) and (f) are from Theorem 5. The upper bound for (g) derives from the conversion from D-1-LAs and the lower bound from the conversion from 2DFAs.

exponential upper bounds and exponential lower bounds. The same questions are open for the simulation of 1-LAs by D-1-LAs and by 2DFAs, namely by dropping the once-marking restriction. We point out that these questions are related to the problem of the cost of the elimination of nondeterminism from two-way finite automata, proposed by Sakoda and Sipser in 1978 [15], which is still open.

5 Always-Marking 1-Limited Automata

Always-marking 1-limited automata replace, when they visit each cell for the first time, the input symbol with its marked version. In this section we study the descriptional complexity of these devices.

First of all, we prove that AM-1-LAs cannot achieve the same succinctness as 1-LAs. In fact, the size gap to 1DFAs reduces from double exponential for 1-LAs to single exponential.

Theorem 6. *Each n-state* AM-1-LA *can be simulated by a* 1NFA *with at most $n \cdot 2^{n^2}$ states and by a complete* 1DFA *with at most $(2^n-1)\cdot 2^{n^2}+1$ states.*

Proof. Let $\mathscr{M} = (Q,\Sigma,\Gamma,\delta,q_0,F)$ be a given n-state AM-1-LA. We adapt the argument used in [8] to convert 1-LAs into 1NFAs and 1DFAs, which is derived from the technique to convert 2DFAs into equivalent 1DFAs, presented in [16], and based on *transitions tables*.

Roughly, transition tables represent the possible behaviors of $\mathscr{M}$ on frozen tape segments. More precisely, given $z \in \Gamma^*$, the *transition table* associated with z is the binary relation $\tau_z \subseteq Q \times Q$, consisting of all pairs (p,q) such that $\mathscr{M}$ has a computation path that starts in the state p on the rightmost symbol of the tape segment containing $\triangleright z$, ends entering the state q by leaving the same tape segment to the right side, i.e., by moving from the rightmost cell of the segment to the right, and does not visit any cell outside the segment.

First, we can apply the conversion presented in [8] from 1-LAs to 1NFAs, in order to obtain from $\mathscr{M}$ an equivalent 1NFA A, whose computations simulate the computations of $\mathscr{M}$ by keeping in the finite state control two components:

- The transition table associated with the part of the tape at the left of the head. This part has been already visited and, hence, it is frozen.
- The state in which the simulated computation of $\mathscr{M}$ reaches the current tape position for the first time.

For details we address the reader to [8, Thm. 2]. Since the number of transition tables is at most 2^{n^2}, the number of states in the resulting 1NFA A is bounded by $n \cdot 2^{n^2}$.

Applying the subset construction, this automaton can be converted into an equivalent deterministic one, with an exponential increase of the number of states, so obtaining a double exponential number of states in n. In the general case, this increasing cannot be reduced. This is due to the fact that different computations of A, after reading the same input, could keep in the control different transitions tables, depending on the fact that $\mathscr{M}$ can replace the same input by different strings.

However, under the restriction we are considering, along different computations, each input string x is always replaced by the same string $\dot{x}$, which is obtained by marking every symbol of x. Hence, at each step of the simulation, the transition table stored by A depends only on the input prefix already inspected. The only part that can change is the state of the simulated computation of $\mathscr{M}$ after reading x.

This allows to obtain from A a 1DFA A', equivalent to $\mathscr{M}$ that, after reading a string x, has in its finite state control the transition table associated with $\dot{x}$, and the *set* of states that the computations of $\mathscr{M}$ can reach after reading x. In other words, the automaton A' is obtained from A by keeping the first component of the control, which is deterministic, and making a subset construction for the second one.

By summarizing, the possible values of the first component are 2^{n^2}, while the values of the second one are 2^n, namely the possible subsets of the state set of $\mathscr{M}$. This gives a $2^n \cdot 2^{n^2}$ upper bound. We can slightly reduce this number, by observing that when the second component contains the empty set, i.e., each computation of $\mathscr{M}$ (or equivalently of A) stops before entering it, then the input is rejected, regardless the first component. Hence, we can replace all the pairs having the empty set as a second component with a unique sink state, so reducing the upper bound to $(2^n - 1) \cdot 2^{n^2} + 1$ □

The asymptotical optimality of the upper bounds in Theorem 6 derives from the optimality of the conversions from 2NFAs to 1NFAs and to 2DFAs [14, 16, 5].

We now show that AM-1-LAs can be more succinct than 2NFAs, even in the deterministic case. In particular we prove the following:

Theorem 7. *The language J_n is accepted by a* D-AM-1-LA *with $O(n)$ states, while it cannot be accepted by any* 2NFA *with less than $2^{\frac{n-1}{2}}$ states.*

Proof. The lower bound for 2NFAs has been given in Theorem 2. The possibility of marking the already-visited cells allows to reduce this cost, even without making use of the nondeterminism, as we now describe. An always marking D-1-LA $\mathscr{M}$ can firstly visit and mark the first n tape cells. Then, it starts to inspect the next block of length n. When the head reaches for the first time a cell, $\mathscr{M}$ remembers the scanned symbol σ in it and moves the head back to the left end-marker and then to the corresponding cell in the first block (this can be implemented with a counter modulo n). If the symbol in this cell is not σ then $\mathscr{M}$ has to skip the remaining symbols in the block under inspection and inspect the next block, if any. This can be done moving the head to the left end-marker and then, starting to count modulo n, moving to the right until finding the first symbol of the next block. This symbol can be located using the value of the counter and the fact that it has not been marked yet. Otherwise, if the symbol in the cell coincides with σ and the block is not completely inspected (see below), $\mathscr{M}$ moves the head to the right to search the next symbol of the block under inspection, namely the first unmarked symbol.

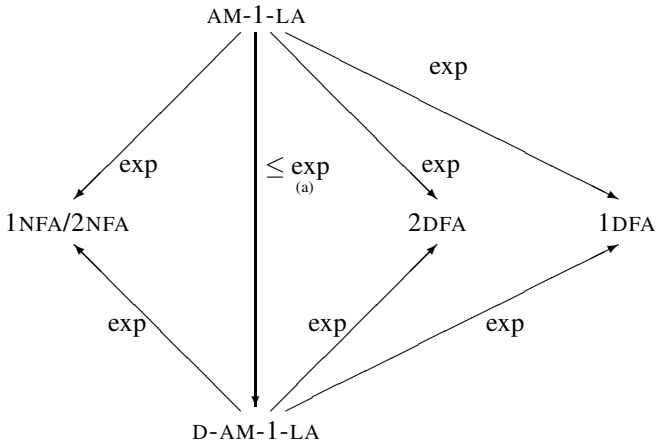

Figure 3: State costs of conversions involving AM-1-LAs. All the exponential upper bounds derive from Theorems 6 and 8, while the lower bounds derive from Theorem 7. For (a) we do not know if in the worst case an exponential size is also necessary.

When locating a symbol, $\mathscr{M}$ can also check and remember if it is in position n. This is useful to detect whether a block has been completely scanned, which also means that the block has been *successfully scanned*, otherwise the machine would have already rejected. Hence, in this case, $\mathscr{M}$ can move the head to the right to finally reach the accepting configuration. However, according to the definition of J_n, before doing that, $\mathscr{M}$ needs to verify that the input has length multiple of n. All these steps can be implemented with a fixed number of variables and a counter modulo n. This allows to conclude that $\mathscr{M}$ can be implemented with $O(n)$ states. □

In Theorem 7 we proved an exponential gap from D-AM-1-LAs to 2NFAs and hence also to one-way finite automata. This allows to conclude that the following upper bounds, that are immediate consequences of the corresponding upper bounds for D-1-LAs [8, Thm. 2], cannot be significantly reduced:

Theorem 8. *Each n-state* D-AM-1-LA *can be simulated by a* 1DFA *and by a* 1NFA *with no more than* $n \cdot (n+1)^n$ *states.*

From the discussion above and Theorem 8, we have the same state gap from D-AM-1-LAs and from D-1-LAs to one-way automata.

The state costs of the conversions involving AM-1-LAs are summarized in Figure 3.

Even in the case of AM-1-LAs, as well as in the cases of 1-LAs and of OM-1-LAs, we do not know how much the elimination of the nondeterminism costs. Here, we have an exponential upper bound for the conversion of AM-1-LAs into D-AM-1-LAs but, at the moment, we do not have a matching lower bound. Considering the conversion of AM-1-LAs into 2DFAs, unlikely the analogous conversions from 1-LAs and OM-1-LAs, here we have matching exponential upper and lower bounds. As already mentioned at the end of Section 4, these questions are related to the open question of Sakoda and Sipser.

6 Conclusion

We study the costs of the simulations of OM-1-LAs and AM-1-LAs by finite automata. Figures 2 and 3 give a summary of the results we obtained. They can be compared with the costs of the simulations

concerning 1-LAs, in Figure 1.

We observed that AM-1-LAs cannot reach the same succinctness as 1-LAs and OM-1-LAs (see Theorems 4 and 6). In particular, in Theorem 3 we have shown that the language K_n can be accepted by a OM-1-LA with $O(n)$ states. Hence, it requires an exponential number of states on AM-1-LAs due to the fact that a double exponential number of states on 1DFAs is necessary (see Theorem 2). It is not difficult to describe a 2NFA accepting K_n with an exponential number of states. We point out that such a machine is also a AM-1-LA. Hence, by summarizing, the language K_n is accepted by a OM-1-LA with $O(n)$ states, by an AM-1-LA with a number of states exponential in n, and by a 1DFA with a number of states double exponential in n. All these costs cannot be reduced.

Since in the nondeterministic case the gaps from OM-1-LAs to finite automata are the same as from 1-LAs, a natural question is to ask if OM-1-LAs are always as succinct as 1-LAs. Intuitively the answer to this question is negative. For instance we do not see how to recognize the language whose strings are concatenations of blocks of length n, in which two blocks are equal, with a OM-1-LA with $O(n)$ states, while it is not hard to accept it using a 1-LA with such a number of states. We leave the study of this question for a future work.

Another candidate for studying this question is the unary language $(a^{2^n})^*$. We proved that this language can be accepted by a D-1-LA with $O(n)$ states and a work alphabet of cardinality $O(n)$, and by a D-1-LA with $O(n^3)$ states and work alphabet of size not dependent on n [10, 12]. As pointed out in [10], each 2NFA accepting it requires at least 2^n states. Hence, by Theorem 5 even each D-OM-1-LA accepting it requires an exponential number of states. We do not see how to reduce this number even by allowing the use of nondeterminism on OM-1-LAs or on AM-1-LAs.

More in general, the comparisons between the sizes of these restricted versions of 1-LAs deserve further investigation, even in the unary case where the cost of several simulations are still unknown [10]. In a recent paper, we investigated *forgetting* 1-LA*s*, another restriction of 1-LAs in which there is a unique symbol X that is used to replace input symbols. Therefore, during the first visit to a cell, its original content is always replaced by X [11].

Finally, we would like to mention once again the problem of the cost of removing nondeterminism from 1-LAs, OM-1-LAs, and AM-1-LAs (see Sections 4 and 5), which is connected to the main question of the cost of the elimination of nondeterminism from two-way finite automata, raised longtime ago by Sakoda and Sipser and still open [15] (for a survey, see [6]).

References

[1] Jean-Camille Birget (1992): *Intersection and Union of Regular Languages and State Complexity*. *Inf. Process. Lett.* 43(4), pp. 185–190, doi:10.1016/0020-0190(92)90198-5.

[2] Viliam Geffert, Carlo Mereghetti & Giovanni Pighizzini (2007): *Complementing two-way finite automata*. *Inf. Comput.* 205(8), pp. 1173–1187, doi:10.1016/j.ic.2007.01.008.

[3] Thomas N. Hibbard (1967): *A Generalization of Context-Free Determinism*. *Inf. Control.* 11(1/2), pp. 196–238, doi:10.1016/S0019-9958(67)90513-X.

[4] John E. Hopcroft & Jeffrey D. Ullman (1979): *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley.

[5] Christos A. Kapoutsis (2005): *Removing bidirectionality from nondeterministic finite automata*. In: *MFCS 2005*, *Lecture Notes in Computer Science* 3618, Springer, pp. 544–555, doi:10.1007/11549345_47.

[6] Giovanni Pighizzini (2013): *Two-Way Finite Automata: Old and Recent Results*. *Fundam. Inform.* 126(2-3), pp. 225–246, doi:10.3233/FI-2013-879.

[7] Giovanni Pighizzini (2019): *Limited Automata: Properties, Complexity and Variants*. In: *DCFS 2019*, *Lecture Notes in Computer Science* 11612, Springer, pp. 57–73, doi:10.1007/978-3-030-23247-4_4.

[8] Giovanni Pighizzini & Andrea Pisoni (2014): *Limited Automata and Regular Languages*. *Int. J. Found. Comput. Sci.* 25(7), pp. 897–916, doi:10.1142/S0129054114400140.

[9] Giovanni Pighizzini & Andrea Pisoni (2015): *Limited Automata and Context-Free Languages*. *Fundam. Inform.* 136(1-2), pp. 157–176, doi:10.3233/FI-2015-1148.

[10] Giovanni Pighizzini & Luca Prigioniero (2019): *Limited automata and unary languages*. *Inf. Comput.* 266, pp. 60–74, doi:10.1016/j.ic.2019.01.002.

[11] Giovanni Pighizzini & Luca Prigioniero (2023): *Forgetting 1-Limited Automata*. In: *NCMA 2023*, Electronic Proceedings in Theoretical Computer Science. To appear. A preliminary version is available at https://doi.org/10.48550/arXiv.2307.16700.

[12] Giovanni Pighizzini & Luca Prigioniero (2023): *Two-way Machines and de Bruijn Words*. In: *CIAA 2023*, *Lecture Notes in Computer Science* 14151, pp. 254–265, doi:10.1007/978-3-031-40247-0_19.

[13] Giovanni Pighizzini, Luca Prigioniero & Simon Šádovský (2022): *1-Limited Automata: Witness Languages and Techniques*. *J. Autom. Lang. Comb.* 27(1-3), pp. 229–244, doi:10.25596/jalc-2022-229.

[14] Michael O. Rabin & Dana S. Scott (1959): *Finite Automata and Their Decision Problems*. *IBM J. Res. Dev.* 3(2), pp. 114–125, doi:10.1147/rd.32.0114.

[15] William J. Sakoda & Michael Sipser (1978): *Nondeterminism and the Size of Two Way Finite Automata*. In: *STOC 1978*, ACM, pp. 275–286, doi:10.1145/800133.804357.

[16] John C. Shepherdson (1959): *The Reduction of Two-Way Automata to One-Way Automata*. *IBM J. Res. Dev.* 3(2), pp. 198–200, doi:10.1147/rd.32.0198.

[17] Michael Sipser (1980): *Halting Space-Bounded Computations*. *Theor. Comput. Sci.* 10, pp. 335–338, doi:10.1016/0304-3975(80)90053-5.

[18] Michael Sipser (1980): *Lower Bounds on the Size of Sweeping Automata*. *J. Comput. Syst. Sci.* 21(2), pp. 195–202, doi:10.1016/0022-0000(80)90034-3.

[19] Klaus W. Wagner & Gerd Wechsung (1986): *Computational complexity*. D. Reidel Publishing Company, Dordrecht.