The Formal Language Theory Column

by

Giovanni Pighizzini

Dipartimento di Informatica
Università degli Studi di Milano
20135 Milano, Italy
pighizzini@di.unimi.it

Regular Languages: To Finite Automata and Beyond

Succinct Descriptions and Optimal Simulations

Luca Prigioniero
Dipartimento di Informatica, Università degli Studi di Milano
via Celoria 18, 20133 Milan, Italy
`prigioniero@di.unimi.it`

Abstract

It is well known that regular (or Type 3) languages are equivalent to finite automata. Nevertheless, many other characterizations of this class of languages in terms of computational devices and generative models are present in the literature. For example, by suitably restricting more general models such as context-free grammars, pushdown automata, and Turing machines, that characterize wider classes of languages, it is possible to obtain formal models that generate or recognize regular languages only. These restricted formalisms provide alternative representations of Type 3 languages that may be significantly more concise than other models that share the same expressing power.

The goal of this work is to provide an overview of old and recent results on these formal systems from a descriptional complexity perspective, that is the study of the relationships between the sizes of such devices. We also present some results related to the investigation of the famous question posed by Sakoda and Sipser in 1978, concerning the size blowups from nondeterministic finite automata to two-way deterministic finite automata.

1 Introduction

The investigation of computational models operating under restrictions is one of classical topics of computer science.

In one of his pioneer papers, Chomsky introduced a hierarchy of classes of languages, also known as *Chomsky hierarchy*, obtained by applying increasing restrictions to general *grammars*, that characterize the class of Type 0 languages [7].

In this way he introduced the classes of *context-sensitive* (or Type 1), *context-free* (or Type 2), and *regular* (or Type 3) languages.

For the same classes of languages, there also exist characterizations in terms of *computational devices*. Even in this case, bounding computational resources of general models, less powerful devices can be obtained. For example, while Turing machines (TM for short), in both deterministic and nondeterministic versions, even in the variant with one tape only, characterize Type 0 languages, by restricting the working space they are allowed to use to the portion of the tape that initially contains the input, linear bounded automata are obtained, that are equivalent to context-sensitive grammars [31]. Also finite automata and pushdown automata (PDA), that are standard recognizers for Type 3 and Type 2 languages, respectively, can be considered as particular Turing machines in which the capacity or access to the memory storage is limited.

Besides the standard models mentioned thus far, considering machines that make restricted use of resources, it is possible to obtain alternative characterization of the classes of the hierarchy. For example, in 1965, Hennie proved that when the length of the computations, *i.e.*, the time, is linear in the input length, one-tape Turing machines are no more powerful than finite automata, that is, they recognize regular languages only [20].

As remarked by Chomsky, context-free languages have the property of being able to describe recursive structures such as, for instance, nested parentheses, arithmetic expressions, and typical programming language constructs. In terms of recognizing devices, this capability is typically implemented through the pushdown store, a memory structure in which the information is stored and recovered in a "last in–first out" way, which is used to add recursion to finite automata, so making the resulting model (pushdown automata), equivalent to context-free grammars [6].

To emphasize the ability of context-free grammars to generate recursive sentential forms, Chomsky investigated the *self-embedding* property [8]: a context-free grammar is self-embedding if it contains some variable which, in some sentential form, is able to reproduce itself enclosed between two nonempty strings. Roughly speaking, this means that such a *self-embedded* variable can generate a "true" recursion that needs an auxiliary memory (typically a stack) to be implemented (in contrast with tail or head recursions, corresponding to the cases in which the two strings surrounding the variable are empty, that can be easily eliminated). Chomsky proved that, among all context-free grammars, only self-embedding ones can generate nonregular languages. Hence, *non-self-embedding grammars* (NSE) are no more powerful than finite automata [7, 8].

Counterpart devices for non-self-embedding grammars, for which the capability of recognizing recursive structures is limited by placing some restrictions on the size of the memory of the corresponding general model, are *constant-height pushdown automata* (h-PDA). More precisely, these devices are standard nondeterministic pushdown automata where the amount of available pushdown store is bounded by some constant $h \in \mathbb{N}$. Hence, the number of their possible configurations is finite, thus implying that they are no more powerful than finite automata.

By contrast to models that make use of space or time restrictions, Hibbard introduced *d-scan limited automata* (or simply *d-limited automata*, d-LA), that are obtained by limiting the writing capabilities of nondeterministic linear bounded automata allowing overwriting of each tape cell only the first d times that it is scanned, for some fixed $d \geq 0$ [21]. Nevertheless, the cell may be visited again and the information stored therein read arbitrarily many more times, but its contents is *frozen* for the remainder of the computation. Hibbard proved that, for each $d \geq 2$ this model characterize context-free languages and showed the existence of an infinite hierarchy of deterministic d-limited automata, whose first level (*i.e.*, corresponding to deterministic 2-limited automata) has been later proved to coincide with the class of *deterministic context-free languages* [40]. (See [32] and references therein for further connections between limited automata and context-free languages.) Furthermore, as shown by Wagner and Wechsung, when $d = 1$, that is, when these devices are allowed to overwrite the contents of each tape cell during the first visit only, 1-limited automata (1-LA) are equivalent to finite automata [53]. Moreover, it is a trivial observation that when $d = 0$, and hence no writings on the tape are allowed, 0-limited automata correspond to finite automata that can move their input head back and forth on the input tape, namely *two-way finite automata*.[1]

General devices and their restrictions characterizing Type 3 languages and discussed in this work are depicted in Figure 1.

Descriptional Complexity of Formal Systems. In this work we shall focus on the models characterizing the bottom level of the Chomsky hierarchy: the class of regular languages. We compare them from a descriptional complexity point of view, specifically, by studying their capability of representing the same class of languages in a more, or less, concise way. More precisely, *descriptional complexity* is a branch of theoretical computer science whose goal is the investigation of the relationships between the sizes of the representations of formal systems that share the same computational power, or, in other words, the study of how concisely a system can describe a class of problems (or languages).

At the beginning of the section a few equivalent models characterizing the class of regular languages have been presented. These devices are obtained by

[1]Further technical details, properties, examples, and references about limited automata can be found in the recent survey by Pighizzini [37].

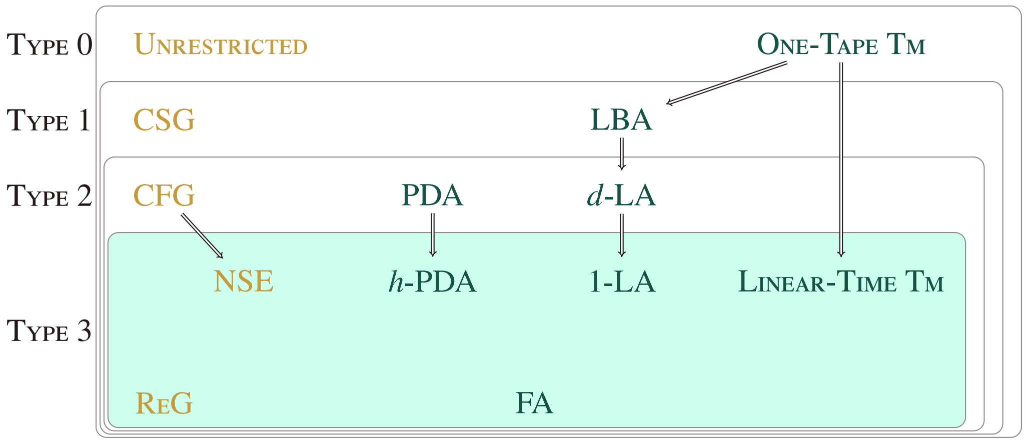

Figure 1: Models representing regular languages obtained by posing some restrictions on standard grammars (on the left, in gold) and recognizers (on the right, in green) characterizing the classes of the Chomsky hierarchy.

limiting some resources of more general models. Therefore, the main question we are interested about is *"How much does the limitation of one resource cost in terms of another resource, or, in other words, what are the upper and lower bounds of such trade-offs?"* [14].

Since we are interested in comparing the sizes of the descriptions of devices and formal systems, for each model under consideration we evaluate its *size* as the total number of symbols used to describe it, or, in other words, to write down its description. In particular, to measure the size of recognizing devices, we consider the amount of memory required to store the "*algorithm*" they implement (their transition function), so, for example, the size of finite automata is bounded by a polynomial in the number of their states. On the other hand, the size of grammars is given by the number of symbols used to write down their derivation rules [30].

Given two different classes of computational models $\mathcal{M}_1$ and $\mathcal{M}_2$ characterizing regular languages, there are natural questions we are interested in:

- Does a function F exist, such that for all the regular languages L, the size of the smallest device of type $\mathcal{M}_1$ for L is upper bounded by the function F of the size of the smallest equivalent device of type $\mathcal{M}_2$? If F exists, it is an *upper bound* for the increase (*blow-up*) in complexity when changing from a minimal model of type $\mathcal{M}_2$ for an arbitrary regular language to an equivalent minimal model of type $\mathcal{M}_1$.
- Do an infinite family of distinct regular languages L_i and a function f exist, such that for all $i \in \mathbb{N}$, the size of the smallest device of type $\mathcal{M}_1$ for L_i is lower bounded by the function f of the size of the smallest equivalent device of type $\mathcal{M}_2$? If f exists, it is a *lower bound* for the increase in complexity

when changing from a minimal model of type $\mathcal{M}_2$ to an equivalent minimal model of type $\mathcal{M}_1$ for infinitely many languages.

If there is no recursive function upper bounding the trade-off between two computational models $\mathcal{M}_1$ and $\mathcal{M}_2$, the trade-off is *non-recursive*. Furthermore, if the lower and the upper bounds coincide, we say that the bound is *optimal*.

For more details about the area of descriptional complexity see, *e.g.*, the surveys by Goldstine et al., Holzer and Kutrib [14, 22].

A classical problem in this field is the investigation of the relationships between deterministic and nondeterministic devices. It is well known that *one-way deterministic* finite automata (1DFA) are sufficient for Type 3 languages. By allowing nondeterministic transitions (1NFAS) the computational power does not increase [48]. A natural question concerning models that share the same computational power is the comparison of their size.

As an example, consider, for any fixed integer $n \geq 0$, the language $L_n = (0+1)^*1(0+1)^{n-1}$, composed by the strings on the alphabet $\{0, 1\}$ whose i-th to last symbol is 1. A 1NFA $\mathcal{A}_n$ could recognize L_n by moving the input head forward on the input tape, until reaching the i-th symbol from the end, that is detected by performing a nondeterministic guess. If such a symbol is 1, the automaton checks whether the length of the remaining part of the input string is $n-1$. Therefore, it is easy to check that the number of states for a NFA accepting L_n is $n+1$. On the other hand, a 1DFA accepting L_n cannot guess which is the i-th symbol from the end of the input string. So, intuitively, it has to "*remember*" a factor representing the last n symbols read, saving it by using the finite control. When the end of the input is reached, the devices checks that the leftmost symbol of the current factor is 1. It is easy to see that the number of the possible factors of length n is 2^n, and each of them is stored by using a state of the 1DFA, thus implying an exponential blowup in states with respect to the equivalent 1NFA.

Therefore, even if deterministic and nondeterministic finite automata characterize the same class of languages, one-way deterministic automata can require exponentially many states with respect to equivalent nondeterministic automata. Hence, there is an exponential size gap from one-way nondeterministic to one-way deterministic automata.

It is also known that even providing finite automata with the capability of moving the input head back and forth along the tape, thus obtaining *two-way* finite automata (2DFA and 2NFA), their recognizing power does not increase. In fact, Shepherdson and, independently, Rabin and Scott, in 1959 proved this result by giving a two constructions that are based on the analysis of the moves of the input head of the automata between the tape cells [50, 48]. Both the constructions, given a two-way finite automaton, return an equivalent one-way deterministic finite automaton whose size is exponential in the square of the size of the automaton given in input.

At this point, one could ask *"Is it possible to exploit the ability to scan the input in a two-way fashion in finite automata to eliminate the nondeterminism?"*. Differently from the one-way case, the question about the size cost of the conversion of (one-way and two-way) nondeterministic finite automata into two-way deterministic finite automata, that was posed by Sakoda and Sipser in 1978, is still open. In their paper, Sakoda and Sipser formulated the problem by the questions

1. For every 2NFA $\mathcal{A}$, does an equivalent 2DFA $\mathcal{A}'$ exist, such that $\mathcal{A}'$ has only polynomially more states than $\mathcal{A}$?
2. For every 1NFA $\mathcal{A}$, does an equivalent 2DFA $\mathcal{A}'$ exist, such that $\mathcal{A}'$ has only polynomially more states than $\mathcal{A}$?

They conjectured that, in both cases, an exponential increase of the size is necessary.

The question has been solved in some special cases that can be grouped in three classes: by considering restrictions on the simulating machines (*e.g.*, *sweeping* [51], *oblivious* [25], *few reversals* two-way deterministic finite automata [26]), by considering restrictions on languages (*e.g.*, *unary case* [12]), by considering restrictions on the simulated machines (*e.g.*, *outer-nondeterministic* automata [10, 28]). However, in spite of all attempts, in the general case the question remains open.

The importance of this open problem is supported by its similarity to the well-known P $\overset{?}{=}$ NP problem [49] and by relationships with the LogSpace $\overset{?}{=}$ NLogSpace question [5, 13, 27, 28]. (See [39] for further details and references.)

Outline. This work is an excerpt of the dissertation [46] and it is organized as follows (a summary of some of the main results obtained is depicted in Figure 2).

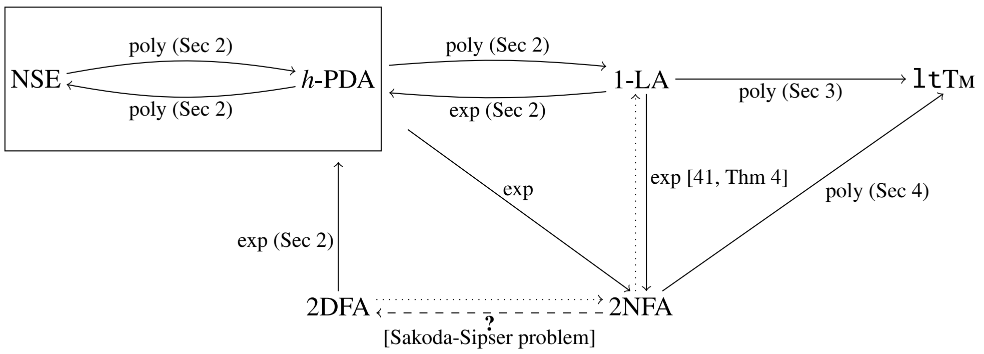

Figure 2: Some of the main results discussed in the work. Dotted arrows denote trivial relationships, while the dashed arrow indicates the Sakoda and Sipser's question [49]. Linear-time Turing machines are denoted `1tTm`.

In Section 2 we present some results on non-self-embedding grammars and their descriptional complexity [43], and compare them with constant-height pushdown automata and 1-limited automata [15].

In Section 3 the time complexity of 1-limited automata [16] is analyzed from a descriptional complexity point of view. Though the model recognizes regular languages only, it may use quadratic time in the input length. With a polynomial increase in size and preserving determinism, each 1-limited automaton can be transformed into a linear-time equivalent one. We also obtained polynomial transformations into related models, including weight-reducing Hennie machines (*i.e.*, one-tape Turing machines syntactically forced to operate in linear-time), and we showed exponential gaps for the converse transformations in the deterministic case.

The investigation about machines working in linear time is continued in Section 4. It is not decidable whether a one-tape Turing machine works in linear time, even if it is deterministic and restricted to use only the portion of the tape which initially contains the input, unless the machine is weight-reducing [17]. By relating the study of Turing machines working in linear time to the above mentioned open question of Sakoda and Sipser, we present the costs of the conversion of nondeterministic finite automata into equivalent linear-time one-tape deterministic machines. A polynomial blowup from two-way nondeterministic finite automata into equivalent weight-reducing one-tape deterministic machines (that work in linear time) was obtained. The blowup remains polynomial if the tape in the resulting machines is restricted to the portion which initially contains the input. However, in this case the resulting machines are not weight-reducing, unless the input alphabet is unary. Similar results are proved in the case the simulated nondeterministic automata are one-way.

In Section 5 we turn our attention to pushdown automata [44]. In particular, we studied pushdown automata without any restriction on the input height and we showed that it cannot be decided whether these devices accept using constant pushdown height, with respect to the input length, or not. Furthermore, in the case of acceptance in constant height, the height cannot be bounded by any recursive function in the size of the description of the machine. In contrast, in the restricted case of pushdown automata over a one-letter input alphabet, *i.e.*, unary pushdown automata, the above property becomes decidable. Moreover, if the height is bounded by a constant that does not depend on the input length, then it is at most exponential with respect to the size of the description of the pushdown automaton. This bound cannot be reduced. Finally, if a unary pushdown automaton uses non-constant height to accept, then the height should grow at least as the logarithm of the input length. This bound is optimal.

In conclusion, in Section 6 we briefly discuss some possible future research directions.

2 Non-Self-Embedding Grammars, Constant-Height Pushdown Automata, and Limited Automata

As mentioned in Section 1, Chomsky investigated the self-embedding property of context-free grammars, and proved that non-self-embedding grammars only generate regular languages. The proof of the result given by Chomsky is constructive: it provides a method for obtaining a finite automaton equivalent to a given non-self-embedding grammar [7, 8]. A different constructive proof of the same result was given by Anselmo, Giammarresi, and Varricchio, who showed that it is possible to decompose non-self-embedding grammars into regular grammars and then to iteratively apply regular substitutions in order to obtain equivalent finite automata [2]. In the same paper, the authors also presented an NSE grammar for the following family of languages. For any fixed integer n, let $U_n = \{ a^{2^n} \}$, be the singleton composed by the unary string of length 2^n. U_n can be generated by an NSE grammar with variables $\{ A_0, \ldots, A_n \}$ such that, for each $i = 1, \ldots, n$, the variable A_i generates two occurrences of the variable A_{i-1} and the variable A_0 produces the terminal a. It is possible to see that the variable A_i derives the string a^{2^i}, for $i = 0, \ldots, n$. Hence, the language generated is U_n. Moreover, the size of the grammar is linear in the parameter n, while it is not hard to verify that the equivalent minimum deterministic finite automaton has $2^n + 1$ states. Actually, as a consequence of state lower bound presented by Mereghetti and Pighizzini [34], the same amount of states are also necessary to accept U_n by using a 2NFA. Therefore, this language witnesses that the size gap between non-self-embedding grammars and equivalent finite automata is at least exponential.

It is worthwhile to mention that, in 1971, Meyer and Fischer proved that for any recursive function f and arbitrarily large integer n, there exists a context-free grammar whose description has size n and which generates a regular language, such that any equivalent finite automaton requires at least $f(n)$ states [35]. This means that it is not possible to obtain a recursive bound relating the size of context-free grammars generating regular languages with the number of states of equivalent deterministic finite automata. It is important to notice that the result of Meyer and Fischer was obtained by considering grammars with a two-letter terminal alphabet. The unary case was studied in 2002 by Pighizzini, Shallit, and Wang, who obtained optimal recursive bounds [45].

We recently proved that also in the case of non-self-embedding grammars the bounds are recursive, independently on the alphabet size [43]. In particular, by inspecting and refining the construction presented by Anselmo, Giammarresi, and Varricchio, we showed that each non-self-embedding grammar of size n can be converted into equivalent nondeterministic and deterministic automata with $2^{O(n)}$ and $2^{2^{O(n)}}$ states, respectively. We also obtained a family of languages that witness that these gaps cannot be reduced. Furthermore, these gaps do not change if we allow the variables which generate only unary strings (*i.e.*, strings consisting of occurrences of only one terminal) to be self-embedded. Such grammars, which are also equivalent to finite automata, are called *quasi-non-self-embedding grammars*.

Other formal models characterizing the class of regular languages and exhibiting gaps of the same order with respect to deterministic and nondeterministic automata are constant-height pushdown automata and 1-limited automata. So, it is natural to study the size relationships between non-self-embedding grammars, constant-height pushdown automata, and 1-limited automata, three models that restrict context-free acceptors to the level of regular recognizers.

The exponential and double exponential gaps from constant-height pushdown automata to nondeterministic and deterministic automata have been proven by Geffert, Mereghetti, and Palano [11]. Furthermore, Bednárová et al. showed the interesting result that the gap from nondeterministic to deterministic constant-height pushdown automata is double exponential also [4]. We remind the reader that both non-self-embedding grammars and constant-height pushdown automata are restrictions of the corresponding general models, where true recursions are not possible. By adapting the standard transformation from PDAS to CFGS, and by modifying a decomposition of NSE grammars presented by Anselmo, Giammarresi, and Varricchio, we proved that non-self-embedding grammars and constant-height pushdown automata are polynomially related in size.

Also 1-limited automata can be significantly smaller than equivalent finite automata. The equivalence between 1-limited automata and finite automata has been investigated from the descriptional complexity point of view by Pighizzini and Pisoni, who proved that each 1-limited automaton $\mathcal{A}$ with n states can be simulated by a one-way deterministic automaton with a number of states double exponential in a polynomial in n. Furthermore, in the worst case, double exponentially many states are necessary for this simulation. The cost reduces to a single exponential when $\mathcal{A}$ is deterministic [41]. The lower bounds in this result have been obtained by providing witness languages defined over a binary alphabet.

We investigated the unary case, for which it was an open question if the same bounds held. In particular, we obtained an exponential gap between unary deterministic 1-limited automata and two-way nondeterministic finite automata [42]. To this aim, we showed that, for each $n > 1$, the singleton language $U_n = \{ a^{2^n} \}$ can be recognized by a deterministic 1-limited automaton having $2n + 1$ states and a description of size $O(n^2)$. Since the same language requires $2^n + 1$ states to be accepted by a one-way nondeterministic automaton, it turns out that the state gap between deterministic 1-limited automata and one-way nondeterministic automata in the unary case is the same as in the binary case. It is an easy observation that the gap does not reduce if we want to convert unary deterministic 1-limited automata into two-way nondeterministic automata.

Such a gap is proven by exploiting an intermediate result on combinatorics on words, which provides a method for constructing a sequence of length 2^n whose elements can be obtained by analyzing the prefix of the sequence obtained until then. This allows the 1-limited automaton recognizing U_n to write the elements of the sequence on the tape by rewriting the contents of the cells during the first visit only, using a number of states and working symbols which is linear in the parameter n.

We also considered the relationships between 1-limited automata and unary context-free grammars. The cost of the conversion of these grammars into finite automata has been investigated and exponential gaps have been proven by Pighizzini, Shallit, and Wang [45]. With the help of a result presented by Okhotin [36], we proved that each unary context-free grammar $\mathcal{G}$ can be converted into an equivalent 1-limited automaton whose description has a size that is polynomial in the size of $\mathcal{G}$ [42].

Let us now turn our attention to the size relationships between 1-limited automata and non-self-embedding grammars.

We obtained a construction transforming each non-self-embedding grammar into a 1-limited automaton of polynomial size. In particular, given an input w, the 1-LA nondeterministically generates a *compression* of a derivation tree of w. Then, it verifies the validity of such a guess. As a consequence, each constant-height pushdown automaton can be transformed into an equivalent 1-limited automaton of polynomial size. Even the conversion of deterministic constant-height pushdown automata into deterministic 1-limited automata costs polynomial in size. For the converse transformation, we showed that an exponential size is necessary. Indeed, consider the following family of languages. For any fixed integer $n > 0$, let P_n be the language of the powers of any string u of length n over the alphabet $\{0, 1\}$, *i.e.*, $P_n = \{u^k \mid u \in \{0, 1\}^n,\ k \geq 0\}$. P_n can be accepted by a two-way deterministic finite automaton (and, hence, by a 1-LA) with $O(n)$ states, but requires exponentially many states to be accepted even by an unrestricted pushdown automaton, which is forced to store the word u in its finite control. From the cost of the conversion of 1-limited automata into nondeterministic automata, it turns out that for the conversion of 1-limited automata into non-self-embedding grammars an exponential size is also sufficient.

Bednárová et al. raised the question of the cost of the conversion of deterministic h-PDAS into 1NFAS [4]. To this regard, it is possible to observe that, for any integer $n \geq 0$, the language $S_n = \left(a^{2^n}\right)^*$ is accepted by a deterministic h-PDA of size polynomial in n (a constant-height pushdown automaton for S_n with a constant number of states, $h = n$, and $2n + 1$ pushdown symbols can be found in [44]) but, as discussed at the beginning of this section, it requires an exponential number of states to be accepted by a finite automaton. Hence, we can conclude that both simulations from two-way automata to h-PDAS and from h-PDAS to two-way automata cost at least exponential.

More details about the results shown in this section can be found in [43] and [15].

3 Limited Automata: A Time Constraint

As observed by Hennie in 1965, deterministic one-tape Turing machines operating in linear time recognize exactly the class of regular languages [20]. The result has later been extended to the nondeterministic case [52, 38]. Here, operating in linear time means that every computation has length linearly bounded in the input length. In particular, linear-time machines are necessarily halting — see [38] for investigations of alternative linear-time restrictions. The above-mentioned result implies that every Hennie machine is equivalent to some finite automaton. From the opposite point of view, this means that providing two-way finite automata with the ability to overwrite the tape cells does not extend the expressiveness of the model, as long as the time is linearly bounded in the length of the input.

However, Průša showed that it is undecidable given a linear bounded deterministic Turing machine to check whether it works in linear time over all input strings, or, in other words, whether it is actually a deterministic Hennie machine [47]. To avoid this drawback, he proposed the weight-reducing variant of Hennie machines, in which the time limitation is syntactic. As a consequence, the number of visits of a cell by the head is bounded by some constant (*i.e.*, not depending on the input length), hence the device works in linear time over every input string.

By contrast to Hennie machines, in d-limited automata the head is allowed to visit a cell after the d-th visit, even if it cannot rewrite the contents anymore. This allows to use super-linear time.

As an example let us consider, for each fixed integer $n \geq 0$, the language

$$Q_n = \left\{ x_0 x_1 \cdots x_k \mid k > 0, \text{ for each } i\colon\ x_i \in \Sigma^n, \text{ for some } j \neq 0\colon\ x_j = x_0 \right\}$$

on the alphabet $\Sigma = \{0, 1\}$. A deterministic 1-LA $\mathcal{A}_n$ may recognize Q_n as follows. It first scans the factor x_0, overwriting each input symbol with a marked copy. Then, $\mathcal{A}_n$ repeats a subroutine which overwrites a factor x_i with the marker $\sharp \notin \Sigma$, while checking whether x_i equals x_0 or not. This can be achieved as follows. Before overwriting the j-th symbol of x_i, first, $\mathcal{A}_n$, with the help of a counter modulo n, moves the head leftward to the position j of x_0 and stores the unmarked scanned symbol σ in its finite control; second, it moves the head rightward until reaching the position j of x_i, namely, the leftmost position that has not been overwritten so far. At this point, $\mathcal{A}_n$ compares the scanned symbol (*i.e.*, the j-th symbol of x_i) with σ (*i.e.*, the j-th symbol of x_0). When $\mathcal{A}_n$ finds out that a complete factor x_i matches x_0, it reaches the end of the input checking that has length multiple of n.

It is possible to implement $\mathcal{A}_n$ with a number of states linear in n and $\#\Sigma + 1$ working symbols. Since for each position of a factor x_i, $i > 0$, the head has to move back to the factor x_0, we observe that $\mathcal{A}_n$ works in quadratic time in the length of the input string.

Therefore, also in the case $d = 1$, 1-limited automata can operate in super-linear time. This contrasts with Hennie machines which operate in linear time by definition. The question we addressed is whether this ability of 1-limited automata with respect to Hennie machines yields a gap between the two models in terms of the size of their representations.

We proved that, operating in super-linear time is not essential for 1-LAS, if allowing a polynomial increase in the number of states. In other words, with a polynomial increase in size, each 1-limited automaton can be transformed into an equivalent linear-time 1-limited automaton, or, alternatively, into a weight-reducing Hennie machine. Furthermore, the obtained device is deterministic when the original machine is deterministic as well. This is achieved by extending the exponential-cost simulation of 1-LA by 2NFA given in [41] (which in turn extends Shepherdson's classic conversion of 2DFAS to 1DFAS [50]) with a method for storing and accessing a carefully chosen subcollection of the many "Shepherdson tables" that a 1NFA would need to remember in its states: the simulating automaton can both store the tables (despite the 1-limitation) and access them efficiently (in both size and time).

We also showed that the 1-limited automata resulting from our constructions have a special structure that can be exploited in order to obtain equivalent 1-limited automata in which an initial phase just overwrites each tape cell, *i.e.*, the device initially performs a nondeterministic left-to-right pass over the tape during which all the cells are independently overwritten. It follows that *reversing* a 1-limited automaton, *i.e.*, transforming it into another one recognizing the reverse of its accepted language, has polynomial cost only. This fails in the deterministic case, for which we exhibit an exponential lower bound. As a consequence, we obtained exponential lower bounds for the simulation of deterministic weight-reducing Hennie machines by deterministic 1-limited automata.

For details about the results discussed in this section please refer to [16].

4 Two-Way Automata and One-Tape Machines

In this section we continue the investigation about devices operating in linear time. In particular, we compare the sizes of descriptions of finite automata with those of equivalent one-tape Turing machines working in linear time. In this section we consider the following variants of one-tape deterministic Turing machines, that are summarized in Figure 3:

Bounded machines. We say that a device is *bounded* if each input string is surrounded by two special symbols called the left and the right endmarkers and the machine is restricted to use only the portion of the tape that initially contains the input (plus the endmarkers), that cannot be left by the head.

Weight-reducing Turing machines. Roughly speaking, in *weight-reducing* Turing machines each overwriting is decreasing with respect to some fixed order on the working alphabet. As a consequence, after overwriting a cell with a minimal symbol, such a machine cannot visit the cell again.

By this condition, in weight-reducing Turing machines the number of visits to each tape cell is bounded by a constant. However, they could have non-halting computations which, hence, necessarily visit infinitely many tape cells.

Linear-time Turing machines. A Turing machine is said to be *linear-time* if, over each input w, its computation halts within $O(|w|)$ steps.

Hennie machines. A Hennie machine is a linear-time Turing machine which is, furthermore, bounded.

Weight-Reducing Hennie machines. These devices are defined by combining previous conditions.

Observe that each bounded weight-reducing Turing machine can execute a number of steps which is at most linear in the length of the input. Consequently, bounded weight-reducing deterministic Turing machines are necessarily Hennie machines.

It is useful to emphasize that, it cannot be decided whether or not a one-tape Turing machine works in linear time.[2] Furthermore, there is no recursive function bounding the size blowup from one-tape Turing machines working in linear time to equivalent finite automata. We proved that these results remain true even in the restricted case of bounded machines.

[2]We mention that it is decidable, though, whether or not a machine makes at most $cm + d$ steps on input of length m, for any *fixed* $c, d > 0$ [9].

$$\begin{tikzcd}
 & \text{one-tape deterministic Turing machines (}\textsc{dTms}\text{)} \arrow[dl, Rightarrow] \arrow[d, Rightarrow] \arrow[dr, Rightarrow] & \\
\text{bounded } \textsc{dTms} \arrow[d, Rightarrow] & \text{linear-time } \textsc{dTms} \arrow[dl, Rightarrow] & \text{weight-reducing } \textsc{dTms} \arrow[ddl, Rightarrow] \\
\text{deterministic Hennie machines} \arrow[dr, Rightarrow] & & \\
 & \text{weight-reducing deterministic Hennie machines} &
\end{tikzcd}$$

Figure 3: Variants of one-tape deterministic Turing machines.

To overcome the above-mentioned "negative" results, we considered weight-reducing machines, that can be seen as a syntactical restriction on one-tape Turing machines. In this case, it can be decided if a deterministic Turing machine is weight reducing. These devices can have non-halting computations. However, they work in linear time as soon as they are halting. In fact, we showed that it is possible to decide whether a weight-reducing machine is halting. As a consequence, it is also possible to decide whether it works in linear time. Moreover, by a polynomial size increase, each weight-reducing machine can be transformed into and equivalent one which always halts and works in linear time. The same blowup is easily extended to weight-reducing Hennie machines. Furthermore, with a polynomial size increase, any weight-reducing machine can be made halting whence working in linear time, while the cost of the simulation by 1DFAS is doubly exponential.

Moreover, each linear-time machine $\mathcal{T}$ can be converted into an equivalent weight-reducing one whose size is bounded by a function of the size and of the execution time of $\mathcal{T}$.

We also related the study of these restricted variants of one-tape machines, accepting only regular languages, to the Sakoda and Sipser question, concerning the size blowups from 1NFAS or 2NFAS to 2DFAS. In this case, as target machines, we considered linear-time one-tape *deterministic* Turing machines. It turned out that each 2NFA $\mathcal{A}$ can be simulated by a one-tape deterministic Turing machine which works in linear time (with respect to the input length) and which has a polynomial size with respect to the size of $\mathcal{A}$. We point out that the resulting machine can use extra space, besides the tape segment which initially contained the input. Next, the machine is halting and weight reducing, thus implying a linear execution time. Hence, nondeterminism can be eliminated with at most a polynomial size increase, obtaining a linear execution time in the input length, and provided the ability to rewrite tape cells and to use some extra space.

We then investigated what happens by removing the latter possibility, namely if the machine does not have any further tape storage, *i.e.*, it is a Hennie machine. We proved that even under this restriction it is still possible to obtain a machine of polynomial size, namely each 2NFA can be transformed into an equivalent Hennie machine of polynomial size. However, the machine resulting from our construction is not weight reducing, unless we require that it agrees with the given 2NFA only on sufficiently long inputs. This problem does not occur in the unary case, where we proved that each unary 2NFA can be simulated by a weight-reducing Hennie machine of polynomial size. Similar results are obtained for the transformation of 1NFAS into variants of one-tape deterministic machines.

The results discussed in this section have been presented in [17].

5 Pushdown Automata and Space Restrictions

As discussed in Section 2, pushdown automata in which the maximum height of the pushdown is limited by some constant, namely constant-height pushdown automata, allow more succinct representations of regular languages than finite automata [11], and are polynomially related in size with their natural generative counterpart, non-self-embedding context-free grammars, roughly, context-free grammars without "true" recursion [7].

In this section we turn our attention on standard pushdown automata, namely with an unrestricted pushdown store, that, however, are able to accept their inputs by making use only of a constant amount of the pushdown store. More precisely, we say that a pushdown automaton $\mathcal{M}$ *accepts in constant height h*, for some given h, if for each word in the language accepted by $\mathcal{M}$ there exists one accepting computation in which the maximum height reached by the store is bounded by h. Notice that this does not prevent the existence of accepting or rejecting computations using an unbounded pushdown height.

It is a simple observation that a pushdown automaton $\mathcal{M}$ accepting in constant height h can be converted into an equivalent constant-height pushdown automaton: in any configuration it is enough to keep track of the current height in order to stop and reject when a computation tries to exceed the height limit. The description of the resulting constant-height pushdown automaton has size polynomial in h and in the size of the description of $\mathcal{M}$.

While studying these size relationships, we tried to understand *how large h can be with respect to the size of the description of $\mathcal{M}$*. We discovered that h can be arbitrarily large. Indeed, adapting an argument presented by Meyer and Fischer to prove non recursive trade-offs between the size of PDAS accepting regular languages and the number of states of equivalent automata [35], we showed that there is no recursive function bounding the maximal height reached by the pushdown store in a pushdown automaton accepting in constant height, with respect to the size of its description. With the same argument, we obtained that there is no recursive function bounding the size blowup from PDAS accepting in constant height to finite automata.

Moreover, using a technique introduced by Hartmanis, based on suitable encodings of single-tape Turing machine computations [19], we also proved that it cannot be decided whether a pushdown automaton accepts in constant height or not.

What does happen, instead, if the language recognized by a pushdown automaton is unary? By studying the structure of the computations of unary pushdown automata, we were able to prove that, in contrast to the general case, it can be decided whether or not they accept in constant height. Furthermore, we proved that if a unary pushdown automaton $\mathcal{M}$ accepts in height h, constant with respect to the input length, then h is bounded by an exponential function in the size of $\mathcal{M}$. By presenting a suitable family of pushdown automata, we showed that this bound cannot be reduced.

Let us turn our attention to pushdown automata that accept using height which is not constant in the input length, in order to investigate how the pushdown height grows. In particular, we asked if there exists a minimum growth of the pushdown height, with respect to the length of the input, when it is not constant. The answer to this question is already known and it derives from results on Turing machines: the height of the store should grow at least as a double logarithmic function [1]. This lower bound cannot be increased, because a matching upper bound has been recently obtained in [3]. As a consequence of the constructions obtained for automata accepting unary languages, we were able to prove that in the unary case this lower bound is logarithmic and it cannot be further increased.

For more details about the results discussed in this section we refer the reader to [44].

6 Future Work

In this work we presented some old and recent results related to the area of descriptional complexity, and, in this regard, we focused on the class of regular languages. To conclude, we discuss possible directions for future research in this field.

First of all, it is worth mentioning that there exist other models characterizing the class of regular languages besides the ones analyzed here. One example are the well-known *regular expressions*, widely discussed in classical textbooks (see, *e.g.* [24]). From regular expressions we can derive a more succinct representation of regular languages, by using *straight line programs*, namely programs representing directed acyclic graphs, whose internal nodes represent the basic regular operations (*i.e.*, union, concatenation, and star). Descriptional complexity of straight line programs has been analyzed and it has been proved that they are polynomially related in size with constant height pushdown automata [11]. Anyways, it would be interesting to deepen the study of descriptional complexity of models *"derived"* from regular expressions.

As widely discussed, the question posed by Sakoda and Sipser in 1978 about the elimination of nondeterminism from two-way automata is still open. We plan to continue the investigation on this question by considering models that have the same computational power of finite automata and by studying the relations in sizes between these nondeterministic devices and finite automata by deterministic ones.

For example, as remarked by Pighizzini in a recent survey, at the moment direct simulations of 1-limited automata by *deterministic* 1-limited automata and by two-way deterministic automata are not known [37]. It could be interesting to study if simulating unary and non-unary 1-limited automata by two-way (instead of one-way) deterministic finite automata the cost reduces from a double exponential to a simple exponential.

It could be also interesting to study "relaxed" versions of the problem of Sakoda and Sipser, in which the simulating machine is a deterministic 1-limited automaton (*i.e.*, a deterministic two-way automaton with the capability of rewriting the contents of tape cells during the first visit).

Moreover, following the research line started in [10], it could be deepen the investigation on the Sakoda and Sipser problem in case of simulated devices that perform a limited use of nondeterminism. In this regard, it is possible to consider several restrictions like, for example, on the number of nondeterministic choices along the computation (see, *e.g.* [23]), or on the number of total, or accepting, computations (also known as *degree of ambiguity* [33, 18, 29]).

Acknowledgment

I am deeply indebted to Giovanni Pighizzini for his supervision during my PhD and his guidance for the writing of my dissertation [46], on which this work is based. I cannot overstate the importance of his advice and support to my development as a researcher and a person.

References

[1] Maris Alberts. "Space Complexity of Alternating Turing Machines". In: *Fundamentals of Computation Theory (FCT) '85*. Vol. 199. Lecture Notes

in Computer Science. Springer, 1985, pp. 1–7. ISBN: 3-540-15689-5. DOI: 10.1007/BFb0028785.

[2] Marcella Anselmo, Dora Giammarresi, and Stefano Varricchio. "Finite Automata and Non-Self-Embedding Grammars". In: *Conference on Implementation and Application of Automata (CIAA) 2002*. Vol. 2608. Lecture Notes in Computer Science. Springer, 2002, pp. 47–56. DOI: 10.1007/3-540-44977-9_4.

[3] Zuzana Bednárová, Viliam Geffert, Klaus Reinhardt, and Abuzer Yakaryilmaz. "New Results on the Minimum Amount of Useful Space". In: *International Journal of Foundations of Computer Science* 27.2 (2016), pp. 259–282. DOI: 10.1142/S0129054116400098.

[4] Zuzana Bednárová, Viliam Geffert, Carlo Mereghetti, and Beatrice Palano. "Removing Nondeterminism in Constant Height Pushdown Automata". In: *Information and Computation* 237 (2014), pp. 257–267.

[5] Piotr Berman and Andrzej Lingas. *On the Complexity of Regular Languages in Terms of Finite Automata*. Tech. rep. 304. Polish Academy of Sciences, 1977.

[6] Noam Chomsky. *Context-Free Grammars and Pushdown Storage*. Tech. rep. 65. Research Laboratory of Electronics: Massachusetts Institute of Technology, 1962, pp. 187–194.

[7] Noam Chomsky. "On Certain Formal Properties of Grammars". In: *Information and Control* 2.2 (1959), pp. 137–167. DOI: 10.1016/S0019-9958(59)90362-6.

[8] Noam Chomsky. "A Note on Phrase Structure Grammars". In: *Information and Control* 2.4 (1959), pp. 393–395. DOI: 10.1016/S0019-9958(59)80017-6.

[9] David Gajser. "Verifying Time Complexity of Turing Machines". In: *Theoretical Computer Science* 600 (2015), pp. 86–97. DOI: 10.1016/j.tcs.2015.07.028.

[10] Viliam Geffert, Bruno Guillon, and Giovanni Pighizzini. "Two-Way Automata Making Choices Only At the Endmarkers". In: *Information and Computation* 239 (2014), pp. 71–86. ISSN: 08905401. DOI: 10.1016/j.ic.2014.08.009.

[11] Viliam Geffert, Carlo Mereghetti, and Beatrice Palano. "More Concise Representation of Regular Languages by Automata and Regular Expressions". In: *Information and Computation* 208.4 (2010), pp. 385–394.

[12] Viliam Geffert, Carlo Mereghetti, and Giovanni Pighizzini. "Converting Two-Way Nondeterministic Unary Automata into Simpler Automata". In: *Theoretical Computer Science* 295.1–3 (2003), pp. 189 –203. ISSN: 0304-3975. DOI: http://dx.doi.org/10.1016/S0304-3975(02)00403-6.

[13] Viliam Geffert and Giovanni Pighizzini. "Two-Way Unary Automata versus Logarithmic Space". In: *Information and Computation* 209.7 (2011), pp. 1016–1025. DOI: 10.1016/j.ic.2011.03.003.

[14] Jonathan Goldstine, Martin Kappes, Chandra M. R. Kintala, Hing Leung, Andreas Malcher, and Detlef Wotschke. "Descriptional Complexity of Machines with Limited Resources". In: *Journal of Universal Computer Science* 8.2 (2002), pp. 193–234. DOI: 10.3217/jucs-008-02-0193.

[15] Bruno Guillon, Giovanni Pighizzini, and Luca Prigioniero. "Non-Self-Embedding Grammars, Constant-Height Pushdown Automata, and Limited Automata". In: *Conference on Implementation and Application of Automata (CIAA) 2018*. Vol. 10977. Lecture Notes in Computer Science. Springer, 2018, pp. 186–197. DOI: 10.1007/978-3-319-94812-6_16.

[16] Bruno Guillon and Luca Prigioniero. "Linear-Time Limited Automata". In: *Theoretical Computer Science* 798 (2019), pp. 95–108. DOI: 10.1016/j.tcs.2019.03.037.

[17] Bruno Guillon, Giovanni Pighizzini, Luca Prigioniero, and Daniel Průša. "Two-Way Automata and One-Tape Machines - Read Only versus Linear Time". In: *Developments in Language Theory (DLT) 2018*. Vol. 11088. Lecture Notes in Computer Science. Springer, 2018, pp. 366–378. DOI: 10.1007/978-3-319-98654-8_30.

[18] Yo-Sub Han, Arto Salomaa, and Kai Salomaa. "Ambiguity, Nondeterminism and State Complexity of Finite Automata". In: *Acta Cybernetica* 23.1 (2017), pp. 141–157. DOI: 10.14232/actacyb.23.1.2017.9.

[19] Juris Hartmanis. "Context-Free Languages and Turing Machine Computations". In: *Mathematical Aspects of Computer Science*. Vol. 19. Proceedings of Symposia in Applied Mathematics. American Mathematical Society, 1967, pp. 42–51.

[20] Fred C. Hennie. "One-Tape, Off-Line Turing Machine Computations". In: *Information and Control* 8.6 (1965), pp. 553–578.

[21] Thomas N. Hibbard. "A Generalization of Context-Free Determinism". In: *Information and Control* 11.1/2 (1967), pp. 196–238.

[22] Markus Holzer and Martin Kutrib. "Descriptional Complexity — An Introductory Survey". In: *Scientific Applications of Language Methods*. Imperial College Press, 2010, pp. 1–58. DOI: 10.1142/9781848165458_0001.

[23] Markus Holzer and Martin Kutrib. “One-Time Nondeterministic Computations”. In: *International Journal of Foundations of Computer Science* 30.6-7 (2019), pp. 1069–1089. doi: 10.1142/S012905411940029X.

[24] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

[25] Juraj Hromkovič and Georg Schnitger. “Nondeterminism versus Determinism for Two-Way Finite Automata: Generalizations of Sipser’s Separation”. In: *International Colloquium on Automata, Languages, and Programming (ICALP) 2003*. Vol. 2719. Lecture Notes in Computer Science. Springer, 2003, pp. 439–451. isbn: 3-540-40493-7. doi: 10.1007/3-540-45061-0_36.

[26] Christos A. Kapoutsis. “Nondeterminism Is Essential in Small Two-Way Finite Automata with Few Reversals”. In: *Information and Computation* 222 (2013), pp. 208–227. doi: 10.1016/j.ic.2012.11.001.

[27] Christos A. Kapoutsis. “Two-Way Automata Versus Logarithmic Space”. In: *Theory of Computing Systems* 55.2 (2014), pp. 421–447. doi: 10.1007/s00224-013-9465-0.

[28] Christos A. Kapoutsis and Giovanni Pighizzini. “Two-Way Automata Characterizations of L/poly Versus NL”. In: *Theory of Computing Systems* 56.4 (2015), pp. 662–685. doi: 10.1007/s00224-014-9560-x.

[29] Chris Keeler and Kai Salomaa. “Nondeterminism Growth and State Complexity”. In: *Descriptional Complexity of Formal Systems (DCFS) 2019*. Vol. 11612. Lecture Notes in Computer Science. Springer, 2019, pp. 210–222. doi: 10.1007/978-3-030-23247-4_16.

[30] Alica Kelemenová. “Complexity of Normal Form Grammars”. In: *Theoretical Computer Science* 28 (1984), pp. 299–314. doi: 10.1016/0304-3975(83)90026-9.

[31] Sige-Yuki Kuroda. “Classes of Languages and Linear-Bounded Automata”. In: *Information and Control* 7.2 (1964), pp. 207–223. doi: 10.1016/S0019-9958(64)90120-2.

[32] Martin Kutrib, Giovanni Pighizzini, and Matthias Wendlandt. “Descriptional Complexity of Limited Automata”. In: *Information and Computation* 259.2 (2018), pp. 259–276. doi: 10.1016/j.ic.2017.09.005.

[33] Hing Leung. “Descriptional Complexity of NFA of Different Ambiguity”. In: *International Journal of Foundations of Computer Science* 16.5 (2005), pp. 975–984. doi: 10.1142/S0129054105003418.

[34] Carlo Mereghetti and Giovanni Pighizzini. “Two-Way Automata Simulations and Unary Languages”. In: *Journal of Automata, Languages and Combinatorics* 5.3 (2000), pp. 287–300.

[35] Albert R. Meyer and Michael J. Fischer. “Economy of Description by Automata, Grammars, and Formal Systems”. In: *Switching Automata Theory (SwAT) 1971*. IEEE Computer Society, 1971, pp. 188–191.

[36] Alexander Okhotin. “Non-erasing Variants of the Chomsky-Schützenberger Theorem”. In: *Developments in Language Theory (DLT) 2012*. Vol. 7410. Lecture Notes in Computer Science. Springer, 2012, pp. 121–129. DOI: 10.1007/978-3-642-31653-1_12.

[37] Giovanni Pighizzini. “Limited Automata: Properties, Complexity and Variants”. In: *Descriptional Complexity of Formal Systems (DCFS) 2019*. Vol. 11612. Lecture Notes in Computer Science. Springer, 2019, pp. 57–73. DOI: 10.1007/978-3-030-23247-4_4.

[38] Giovanni Pighizzini. “Nondeterministic One-Tape Off-Line Turing Machines”. In: *Journal of Automata, Languages and Combinatorics* 14.1 (2009), pp. 107–124.

[39] Giovanni Pighizzini. “Two-Way Finite Automata: Old and Recent Results”. In: *Fundamenta Informaticae* 126.2-3 (2013), pp. 225–246. DOI: 10.3233/FI-2013-879.

[40] Giovanni Pighizzini and Andrea Pisoni. “Limited Automata and Context-Free Languages”. In: *Fundamenta Informaticae* 136.1-2 (2015), pp. 157–176. DOI: 10.3233/FI-2015-1148.

[41] Giovanni Pighizzini and Andrea Pisoni. “Limited Automata and Regular Languages”. In: *International Journal of Foundations of Computer Science* 25.7 (2014), pp. 897–916. DOI: 10.1142/S0129054114400140.

[42] Giovanni Pighizzini and Luca Prigioniero. “Limited Automata and Unary Languages”. In: *Information and Computation* 266 (2019), pp. 60–74. DOI: 10.1016/j.ic.2019.01.002.

[43] Giovanni Pighizzini and Luca Prigioniero. “Non-Self-Embedding Grammars and Descriptional Complexity”. In: *Non-Classical Models of Automata and Applications (NCMA) 2017*. 2017, pp. 197–209.

[44] Giovanni Pighizzini and Luca Prigioniero. “Pushdown Automata and Constant Height: Decidability and Bounds”. In: *Descriptional Complexity of Formal Systems (DCFS) 2019*. Vol. 11612. Lecture Notes in Computer Science. Springer, 2019, pp. 260–271. DOI: 10.1007/978-3-030-23247-4_20.

[45] Giovanni Pighizzini, Jeffrey O. Shallit, and Ming-wei Wang. “Unary Context-Free Grammars and Pushdown Automata, Descriptional Complexity and Auxiliary Space Lower Bounds”. In: *Journal of Computer and System Sciences* 65.2 (2002), pp. 393–414.

[46] Luca Prigioniero. “Regular Languages: To Finite Automata and Beyond - Succinct Descriptions and Optimal Simulations”. PhD thesis. Università degli Studi di Milano, Dipartimento di Informatica, Jan. 2020.

[47] Daniel Průša. “Weight-Reducing Hennie Machines and Their Descriptional Complexity”. In: *Language and Automata Theory and Applications (LATA) 2014*. Vol. 8370. Lecture Notes in Computer Science. 2014, pp. 553–564.

[48] Michael O. Rabin and Dana Scott. “Finite Automata and Their Decision Problems”. In: *IBM Journal of Research and Development* 3.2 (1959), pp. 114–125. DOI: 10.1147/rd.32.0114.

[49] William J. Sakoda and Michael Sipser. “Nondeterminism and the Size of Two Way Finite Automata”. In: *Symposium on Theory of Computing (SToC) 1978*. ACM, 1978, pp. 275–286. DOI: 10.1145/800133.804357.

[50] John C. Shepherdson. “The Reduction of Two-Way Automata to One-Way Automata”. In: *IBM Journal of Research and Development* 3.2 (1959), pp. 198–200. DOI: 10.1147/rd.32.0198.

[51] Michael Sipser. “Halting Space-Bounded Computations”. In: *Theoretical Computer Science* 10.3 (1980), pp. 335–338. ISSN: 0304-3975.

[52] Kohtaro Tadaki, Tomoyuki Yamakami, and Jack C. H. Lin. “Theory of One-Tape Linear-Time Turing Machines”. In: *Theoretical Computer Science* 411.1 (2010), pp. 22–43. DOI: 10.1016/j.tcs.2009.08.031.

[53] Klaus W. Wagner and Gerd Wechsung. *Computational Complexity*. Dordrecht: D. Reidel Publishing Company, 1986.