UNARY AUTOMATA SIMULATIONS AND CYCLIC LANGUAGES [1]

Carlo Mereghetti and Giovanni Pighizzini
Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano
via Comelico 39
20135 Milano – Italy
e-mail: {mereghc,pighizzi}@dsi.unimi.it

ABSTRACT

A tight lower bound on the number of states of two–way deterministic and nondeterministic automata accepting unary languages is proved. This result represents a useful tool to study the cost — in terms of states — of the simulations of two–way automata with other kinds of automata and vice versa.

We also show that any one–way nondeterministic automaton accepting a *unary cyclic* language can be simulated by a two–way deterministic automaton without increasing the number of states. This improves the quadratic simulation known for unary languages. Furthermore, by using the above mentioned lower bound, we are able to show that our simulation is optimal.

Keywords: Unary automata, cyclic automata, two–way automata, descriptional complexity

1. Introduction

Several features have been added to the original notion of *one–way deterministic* finite automaton (1dfa, for short) [9]. Thus, we saw one–way nondeterministic (1nfa) and alternating automata, and yet the ability of scanning input strings back and forth, thus yielding the definition of *two–way* devices (e.g., 2dfa, 2nfa). Simulation results have shown that all these new machines share the same computational power with the original one: they all exactly characterize the class of *regular languages*.

Even if this would seem to close the story, a lot of works in the literature have tackled such equivalence issues from a descriptional complexity point of view. Indeed, studying how the number of states changes when turning one automaton into another gives an idea on how succinctly we can represent regular languages by using, e.g., nondeterminism and/or two–way motion of the input head. It is well–known, for instance, that 1nfa's can be logarithmically more succinct than 1dfa's, and that also two–way scanning can lead to logarithmically smaller devices. We address the interested reader to [1] which is a valuable source for results and references.

It is worth noticing that some questions concerning the true expressivity power of automata are still open, the most important being probably that posed by W. Sakoda and M. Sipser in [11]: how many states are necessary and sufficient to simulate 2nfa's or 1nfa's with 2dfa's? They conjecture such a cost to be exponential, and Sipser [10] proves that this is exactly the case when 2dfa's are required to be *sweeping*, i.e., to have head reversals only at the ends of the input tape. P. Berman and A. Lingas [2] state a lower bound of $\Omega(n^2/\log n)$ for the general problem, and provide an interesting connection with the celebrated open problem Dlogspace $\stackrel{?}{=}$ Nlogspace. This might give added evidence that studying simulation costs between automata is not only interesting *per se*, but it is also related to fundamental problems in complexity.

Several attempts have been made to solve "simpler" versions of the Sakoda–Sipser's open problem, for instance, by focusing on restricted type of automata. To this regard, one of the most promising research line goes through the study of optimal simulations between *unary automata*, i.e., automata with a single

[1] Partially supported by MURST, under the project "Modelli di calcolo innovativi: metodi sintattici e combinatori".

letter input alphabet. The problem of evaluating the costs of unary automata simulations was raised in [10], and has leaded to emphasize some relevant differences with the general case. For instance, we know that $O(e^{\sqrt{n \log n}})$ states suffice in order to simulate a unary n–state 1nfa or 2dfa by a 1dfa. Furthermore, a unary n–state 1nfa can be simulated by a 2dfa having $O(n^2)$ many states, and this closes the 1nfa vs. 2dfa Sakoda–Sipser's open problem, at least in the unary case. All these results and their optimality have been proved in 1986 by M. Chrobak [3]. Further insights into the unary world have been given in [8]. Specifically, authors prove that $O(e^{\sqrt{n \log n}})$ is also the optimal cost of simulating 2nfa's via 1dfa's. Moreover, they show that, by just paying a quadratic increment in the number of states, unary 2nfa's can always be regarded to as having both reversals and nondeterministic choices *only* at the ends of the input. This may shed some light on the Sakoda–Sipser's open question. Figure 1 summarizes some of the optimal simulation costs of unary automata obtained in [3, 8].

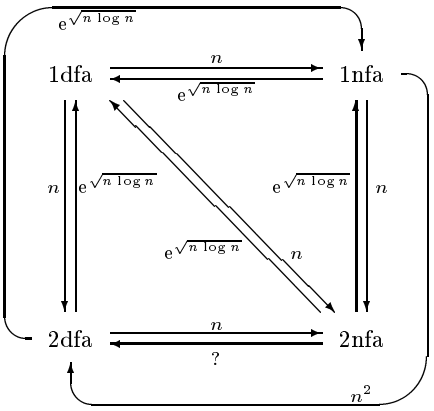

Figure 1: Costs of the optimal simulations between different kinds of unary automata [3, 8]. An arc labeled $f(n)$ from a vertex x to a vertex y means that a unary n–state automaton of type x can be simulated by a $O(f(n))$–state automaton of type y. The arc labeled "?" is the Sakoda–Sipser's open question.

In this work, we further analyze the simulations between unary automata, emphasizing the relevant role of *unary cyclic languages*. Basically, for a positive integer λ, a unary λ–cyclic language is a unary language that can be accepted by a 1dfa whose transition graph is a cycle of length λ. This kind of automata have been considered in [5], and often appear when studying unary regular languages. For instance, it is well–known that each unary regular language differs from a cyclic language for a finite number of strings. Using this property, the quadratic simulation in [8] that transforms a unary 2nfa into a 2nfa having both reversals and nondeterminism only at the ends of the input, basically reduces to checking membership into a cyclic language. As a consequence, to solve the Sakoda–Sipser's open problem in the unary case, it is enough to solve it in the more manageable realm of cyclic languages.

The main result of the paper is a tight lower bound on the number of states of 2nfa's. More precisely, we show *that each 2nfa accepting a properly definitively λ–cyclic language* (essentially, a unary language whose minimum 1dfa contains a cycle of λ states), *where λ factorizes as $\lambda = p_1^{k_1} \cdot p_2^{k_2} \cdot \ldots \cdot p_s^{k_s}$, cannot employ less than $p_1^{k_1} + p_2^{k_2} + \cdots + p_s^{k_s}$ states.* This result, which strengthens the analogous one obtained in [5] for 1nfa's, relates the structure of a 2nfa to numerical properties of the accepted language. Hence, it represents a useful tool to study simulations of 2nfa's with other kind of automata and vice versa. It is worth remarking that we prove the optimality of the lower bound even for 2dfa's.

As a further result, we show that each n–state 1nfa A accepting a unary cyclic language can be transformed into a $(n+1)$–state 1nfa A' in *simple normal form* (a notion introduced in [5]). This enables us to give a precise answer to the 1nfa vs. 2dfa Sakoda–Sipser's open problem for unary automata that accept cyclic languages. In fact, we prove that *each unary n–state 1nfa accepting a cyclic language can be simulated by a 2dfa with no more than n states.* The optimality proof of this simulation is a first example of an application of our main result. We point out that simulating general unary n–state 1nfa's with 2dfa's costs n^2 many states, as one may read from Figure 1.

We conclude the paper by addressing some optimality issues, and open problems.

2. Preliminary notions and results

Given a set S, $\sharp S$ denotes its cardinality, and 2^S the family of all its subsets. A language $\mathscr{L}$ is said to be *unary* (or *tally*) whenever it can be built over a *single letter* alphabet. In this case, we let $\mathscr{L} \subseteq 1^*$.

As usual, we let $\mathbf{Z}^+$ be the set of positive integers. For any positive number z, $\log z$ denotes the natural logarithm of z. The Fundamental Theorem of Arithmetics says that each $z \in \mathbf{Z}^+$ can be univocally expressed as a product $z = p_1^{k_1} \cdot p_2^{k_2} \cdot \ldots \cdot p_s^{k_s}$, where $p_1 < p_2 < \cdots < p_s$ are s primes, and $k_1, k_2, \ldots k_s \in \mathbf{Z}^+$. By the Prime Number Theorem, we know that $s = O(\frac{z}{\log z})$. The *greatest common divisor* of integers $a_1, \ldots, a_s$ writes as $\gcd(a_1, \ldots, a_s)$. Their *least common multiple* is denoted by $\mathrm{lcm}(a_1, \ldots, a_s)$. In estimating the simulation costs of unary automata, a crucial role is played by the function

$$F(n) = \max\{\mathrm{lcm}(x_1, \ldots, x_s) \mid x_1, \ldots, x_s \in \mathbf{Z}^+ \text{ and } x_1 + \cdots + x_s = n\}.$$

Evaluating the growth rate of $F(n)$ is known as Landau's problem [6, 7]. To our purposes, it is enough to know that

Lemma 2.1 $F(n) = O(\mathrm{e}^{\sqrt{n \log n}})$.

Let us take a brief look on the computational model of finite automata. For a detailed exposition, we refer the reader to [4]. A *one–way nondeterministic finite automaton* (1nfa) is a 5–tuple $A = (Q, \Sigma, \delta, q_0, F)$, where Q is the finite set of states, Σ is the finite input alphabet, $\delta : Q \times \Sigma \to 2^Q$ is the transition function, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states. The transition function δ can be extended to strings in a standard way. The language accepted by A is the set $\mathscr{L}(A) = \{x \in \Sigma^* \mid \delta(q_0, x) \cap F \neq \emptyset\}$. The one–way automaton A is *deterministic* (1dfa) if and only if $\sharp\delta(q, \sigma) \leq 1$, for any $q \in Q$, and $\sigma \in \Sigma$.

It is well–known that a one–way automaton can be viewed as a control unit that reads, via an input head, a tape containing input strings stored one symbol per square. At each move, the input head is shifted one square right. This model can be extended to encompass *stationary* and *left* moves of the input head. More precisely, we can define a *two–way nondeterministic finite automaton* (2nfa) as a 5–tuple $A = (Q, \Sigma, \delta, q_0, F)$ in which Q, Σ, q_0, and F are defined as for 1nfa's, while $\delta : Q \times (\Sigma \cup \{\vdash, \dashv\}) \to 2^{Q \times \{-1, 0, +1\}}$ is the transition function, where $\vdash, \dashv$ are two special symbols — the left and the right endmarker, respectively — not in Σ. Input strings are stored onto the input tape surrounded by the two endmarkers, the left endmarker being at the 0th square. The computation begins with the input head scanning the 1st square. In a move, A reads an input symbol, changes its state, and moves the input head one position forward, backward, or keeps it stationary depending on whether δ returns $+1$, -1, or 0, respectively. Left and right moves on the left and right endmarker, respectively, are forbidden. Two–way deterministic automata (2dfa) are defined in the obvious way. Without loss of generality, we stipulate that a two–way automaton accepts its input if and only if it reaches a final state with the input head scanning one of the endmarkers.

We call *unary* any automaton that works with a single letter input alphabet. A unary 1dfa is said to be *cyclic* if and only if its transition digraph is simply a unique (chordless) cycle. As we will see later on, cyclic automata recognize cyclic languages which are defined as follows:

Definition 2.1 *A unary language $\mathscr{L}$ is* cyclic *whenever there exists a positive integer λ such that*

$$1^n \in \mathscr{L} \text{ if and only if } 1^{n+\lambda} \in \mathscr{L},$$

for any $n \geq 0$. To emphasize the periodicity of $\mathscr{L}$, we say that $\mathscr{L}$ is λ–cyclic. *$\mathscr{L}$ is* properly λ–cyclic *if and only is it is λ–cyclic, but not λ'–cyclic, for any $\lambda' < \lambda$.*

The following properties can be trivially proved:

Theorem 2.1 *Given a unary language $\mathscr{L}$ and a positive integer λ, the following properties hold:*

(i) $\mathscr{L}$ is λ–cyclic if and only if it is recognized by a cyclic 1dfa with λ states.

(ii) $\mathscr{L}$ is properly λ–cyclic if and only if χ is the number of states of the minimum 1dfa accepting $\mathscr{L}$.

It is well–known that (infinite) unary regular languages form *definitively periodic sets*, as stated in the following

Theorem 2.2 *Let $\mathscr{L}$ be a unary regular language. Then, there exist two integers $\nu, \lambda \geq 0$ such that, for any $n > \nu$,*

$$1^n \in \mathscr{L} \text{ if and only if } 1^{n+\lambda} \in \mathscr{L}.$$

To emphasize the periodicity, a language $\mathscr{L}$ as in Theorem 2.2 is said to be *definitively λ–cyclic*. $\mathscr{L}$ is *properly definitively λ–cyclic* if and only is it is definitively λ–cyclic, but not definitively λ'–cyclic, for any $\lambda' < \lambda$.

3. A lower bound on the size of 2nfa's

In this section, we state a tight lower bound on the number of states of 2nfa's accepting unary languages, which is related to the periodicity of the accepted language. Such a lower bound improves a result stated in [5, Cor. 2.1] for 1nfa's accepting cyclic languages.

Theorem 3.1 *Let $\mathscr{L}$ be a unary language which is definitively properly λ–cyclic, with $\lambda = p_1^{k_1} \cdot p_2^{k_2} \cdot \ldots \cdot p_s^{k_s}$. Then, the number of states of each 2nfa accepting $\mathscr{L}$ is at least $p_1^{k_1} + p_2^{k_2} + \cdots + p_s^{k_s}$.*

Proof. Given a unary 2nfa accepting $\mathscr{L}$, by Theorem 10 in [8], there exists a set of positive integers $\{\ell_1, \ldots, \ell_r\} \subseteq \{1, \ldots, n\}$ such that $\ell_1 + \cdots + \ell_r \leq n$, and $\mathscr{L}$ is definitively $\tilde{\lambda}$–cyclic, where $\tilde{\lambda} = \mathrm{lcm}(\ell_1, \ldots, \ell_r)$. It is not difficult to observe that either $\lambda = \tilde{\lambda}$, or λ divides $\tilde{\lambda}$, i.e,

$$\tilde{\lambda} = \mathrm{lcm}(\ell_1, \ldots, \ell_r) = k \cdot p_1^{k_1} \cdot p_2^{k_2} \cdot \ldots \cdot p_s^{k_s},$$

for some $k \geq 1$. Hence, each $p_j^{k_j}$ divides some of ℓ_i's, and then

$$n \geq \ell_1 + \ell_2 + \cdots + \ell_r \geq p_1^{k_1} + p_2^{k_2} + \cdots + p_s^{k_s}.$$

□

We point out that if λ is a prime power, namely $\lambda = p_1^{k_1}$, then, from Theorem 3.1, each 2nfa accepting a language $\mathscr{L}$ which is properly λ–cyclic must have at least λ states. But this is exactly the number of states of the minimum 1dfa accepting $\mathscr{L}$. Thus, in this case, the descriptions of λ–cyclic languages in terms of 2nfa's, 1nfa's, 2dfa's, and 1dfa's, have the same cost.

Now, we prove the optimality of the lower bound stated in Theorem 3.1 even for 2dfa's, in the general case, namely, for each integer λ:

Theorem 3.2 *For each integer λ, with $\lambda = p_1^{k_1} \cdot p_2^{k_2} \cdot \ldots \cdot p_s^{k_s}$, there exists a properly λ–cyclic language $\mathscr{L}_\lambda$ which is accepted by a 2dfa with $p_1^{k_1} + p_2^{k_2} + \cdots + p_s^{k_s}$ states.*

Proof. (outline) Consider the language $\mathscr{L}_\lambda = \{1^{\lambda k} \mid k \geq 0\}$. It is immediate to verify that $\mathscr{L}_\lambda$ is properly λ–cyclic. In order to recognize $\mathscr{L}_\lambda$, a 2dfa A performs s input scans: in the ith pass, using $p_i^{k_i}$ states, A checks whether or not $p_i^{k_i}$ divides the input length. □

In Figure 2, a 2dfa built as in Theorem 3.2, and accepting the language $\mathscr{L}_{30}$ is represented.

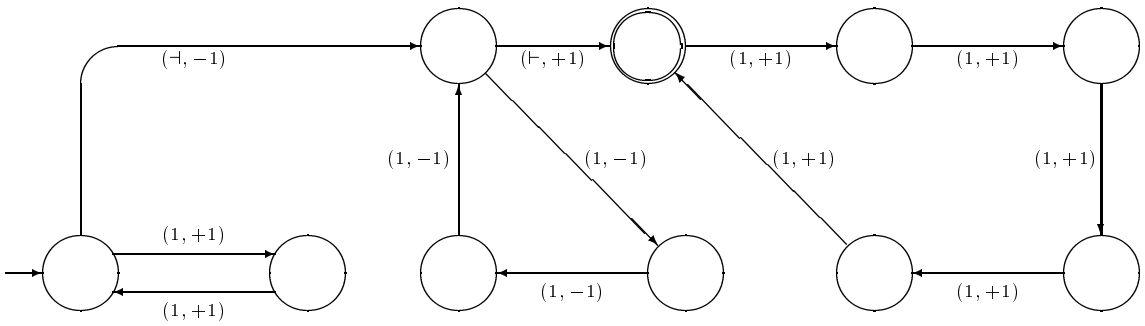

Figure 2: A 2dfa accepting the language $\mathscr{L}_{30}$.

As a consequence of Theorem 3.1 and of Theorem 3.2, we get that the trivial simulation of 2dfa's by 2nfa's cannot be improved.

In the case of integers λ which are products of two prime powers, i.e., $\lambda = p_1^{k_1} \cdot p_2^{k_2}$, Theorem 3.2 can be strengthened, at least for nondeterministic automata, by showing that *each* λ–cyclic regular language can be accepted by a 2nfa with $p_1^{k_1} + p_2^{k_2}$ states:

Theorem 3.3 *Let $\mathscr{L}$ be a λ–cyclic language, with $\lambda = p_1^{k_1} \cdot p_2^{k_2}$. Then, there exists a 2nfa A with $p_1^{k_1} + p_2^{k_2}$ states which recognizes $\mathscr{L}$.*

Proof. Let $X = \{0 \leq n \leq \lambda - 1 \mid 1^n \in \mathscr{L}\}$. Each integer $x \in X$ can be represented as the pair (x', x''), with $x' = x \bmod p_1^{k_1}$ and $x'' = x \bmod p_2^{k_2}$.

The automaton A consists of two cycles of lengths $p_1^{k_1}$ and $p_2^{k_2}$, respectively. Given an input of length n, A performs a first scan, using the first cycle, to compute $n' = n \bmod p_1^{k_1}$. Then, an integer n'' is nondeterministically choosen, such that the pair (n', n'') represents an element belonging to X, if any. In a second scan, starting from the state $p_2^{k_2} - n''$ of the second cycle, the automaton verifies whether or not $n'' = n \bmod p_2^{k_2}$. □

In Figure 3, a 2nfa built as in Theorem 3.3, and accepting $\mathscr{L} = \{1^n \mid n \bmod 6 \leq 2\}$ is represented.

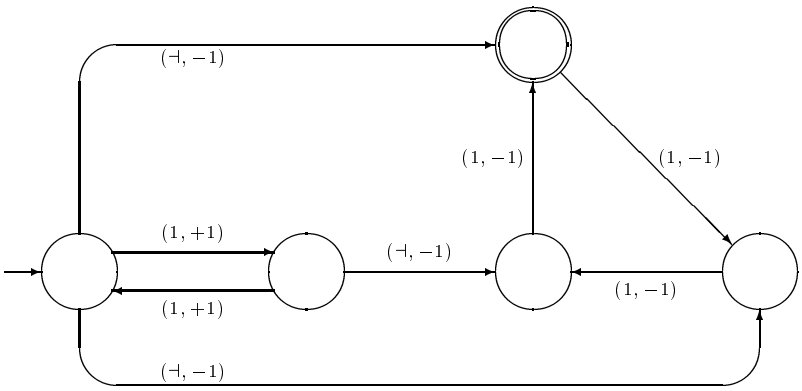

Figure 3: A 2nfa accepting the language $\mathscr{L} = \{1^n \mid n \bmod 6 \leq 2\}$.

At the moment, we do not known whether Theorem 3.3 holds for 2dfa's as well. However, we point out that the minimum 1dfa accepting a λ–cyclic language has λ states and, in the case of $\lambda = p_1^{k_1} \cdot p_2^{k_2}$, this number is bounded by $(p_1^{k_1} + p_2^{k_2})^2$. More generally, by combining the lower bound stated in Theorem 3.1 with this observation, one can easily conclude that each n–state 2nfa accepting a λ–cyclic language, with $\lambda = p_1^{k_1} \cdot p_2^{k_2} \cdot \ldots \cdot p_s^{k_s}$ and $n = p_1^{k_1} + p_2^{k_2})^2 + \cdots + p_s^{k_s}$, can be simulated by a 1dfa whose number of states is polynomial in n and exponential in s.

4. Normal forms for unary 1nfa's and simulation by 2dfa's

We begin by recalling the *Chrobak normal form* [3], and the *simple normal form* [5] for unary 1nfa's. We show that for each n–state 1nfa accepting a unary cyclic language, there exists an equivalent $(n+1)$–state 1nfa in simple normal form. This enables us to prove that any n–state 1nfa accepting a unary cyclic language can be optimally simulated by an n–state 2dfa.

Definition 4.1 [3, Def. 4.2] *A unary 1nfa $A = (Q, \{1\}, \delta, q_0, F)$ is in* Chrobak normal form *if Q can be partitioned into $k+1$ disjoint sets $Q_0, C_1, \ldots, C_k$ such that:*

- $Q_0 = \{q_0, q_1, \ldots, q_m\}$,
- *for* $i = 1, 2, \ldots, k$, $C_i = \{p_{i,0}, p_{i,1}, \ldots, p_{i,y_i-1}\}$,
- *for* $i = 1, 2, \ldots, k$, *and* $j = 0, 1, \ldots, y_i - 1$, $\delta(p_{i,j}, 1) = \{p_{i,(j+1) \bmod y_i}\}$, *i.e., C_i is a cycle of length y_i,*
- *for* $i = 0, 1, \ldots, m-1$, $\delta(q_i, 1) = \{q_{i+1}\}$,
- $\delta(q_m, 1) = \{p_{1,0}, p_{2,0}, \ldots, p_{k,0}\}$.

If Q_0 consists of a single state, then the automaton A is said to be in simple normal form.

In Figure 4, a 1nfa in simple normal form is displayed.

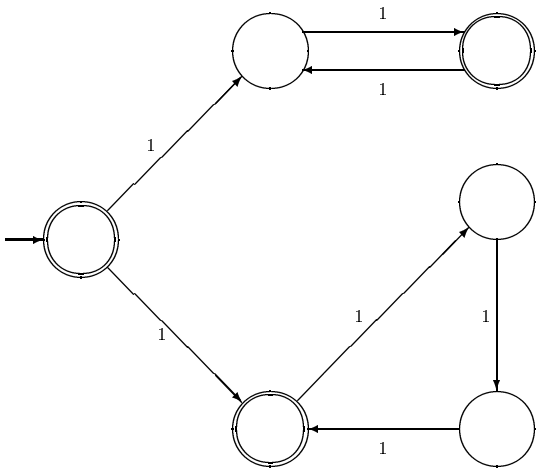

Figure 4: A 1nfa in simple normal form.

In [3, Lemma 4.3], it is proved that each unary n–state 1nfa can be simulated by an equivalent $O(n^2)$–state 1nfa in Chrobak normal form *with at most n states in the cycles.* Here, we improve this result in the case of 1nfa's accepting unary cyclic languages.

Theorem 4.1 *For each unary n–state 1nfa accepting a cyclic language $\mathscr{L}$, there exists an equivalent 1nfa in simple normal form which has no more than $n+1$ states.*

Proof. (outline) Given an n–state 1nfa, by the result in [3, Lemma 4.3] mentioned before this theorem, we can get an equivalent 1nfa $A = (Q, \{1\}, \delta, q_0, F)$ in Chrobak normal form, with at most n states in the cycles, i.e., using notation in Definition 4.1, with $\sharp(C_1 \cup C_2 \cup \cdots \cup C_k) = \sum_{i=1}^{k} y_i \leq n$. From A, we construct a 1nfa A' in simple normal form by suitably "rolling" up the states $q_1, q_2, \ldots, q_m$ onto cycles $C_1, C_2, \ldots, C_k$. In this, we use the fact that A recognizes a cyclic language. More formally, we let $A' = (Q', \{1\}, \delta', q_0, F')$, where:

- $Q' = \{q_0\} \cup C_1 \cup \cdots \cup C_k = Q - \{q_1, q_2, \ldots, q_m\}$,
- $F' = F - \{q_1, q_2, \ldots, q_m\}$,
- $\delta'(q_0, 1) = \{p_{1,(-m) \bmod y_1}, p_{2,(-m) \bmod y_2}, \cdots, p_{k,(-m) \bmod y_k}\}$.

It is not hard to prove that the automata A and A' are equivalent. □

Now, we are ready to simulate 1nfa's with 2dfa's. In [3, Thm. 4.4], it is proved that each unary n–state 1nfa can be simulated by a $O(n^2)$–state 2dfa. Moreover, this simulation is optimal. Using Theorem 4.1, we are able to reduce the cost of this simulation in the case of 1nfa's accepting cyclic languages.

Theorem 4.2 *Each unary n–state 1nfa A accepting a cyclic language can be simulated by a 2dfa B with at most n states.*

Proof. By Theorem 4.1, A can be simulated by a $(n+1)$–state 1nfa A' in simple normal form, containing k cycles $C_1, C_2, \ldots, C_k$. Thus, on any input, B makes at most k scans. In the ith scan, it simulates the transitions of the cycle C_i of A (besides the transition from the initial state). B accepts whenever a scan ends in a final state. □

In Figure 5, the 2dfa obtained by applying to the 1nfa in simple normal form of Figure 4 the construction given in the proof of Theorem 4.2 is represented.

In order to prove that the simulation represented in Theorem 4.2 is tight (at least for those automata whose number of states is the sum plus 1 of prime powers that are relatively primes), we now make use of the lower bound stated in Theorem 3.1.

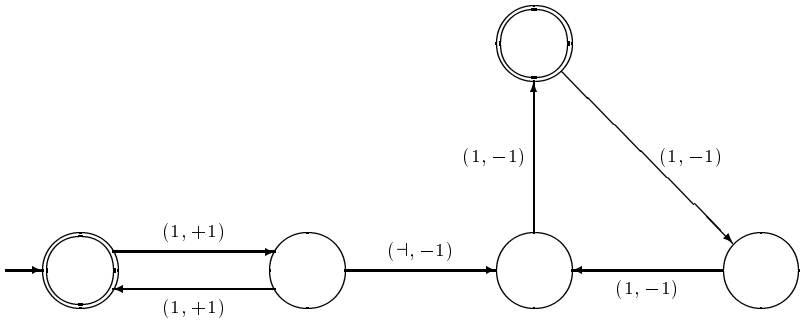

Figure 5: A 2dfa obtained as in Theorem 4.2.

Theorem 4.3 *Given an integer* $\lambda = p_1^{k_1} \cdot p_2^{k_2} \cdot \ldots \cdot p_s^{k_s}$, *and a 1nfa with* $n = p_1^{k_1} + p_2^{k_2} + \cdots + p_s^{k_s} + 1$ *states, there exists a language accepted by a 1nfa with* n *states for which any 2nfa requires* $n - 1$ *states.*

Proof. The properly λ–cyclic language $\mathscr{L}'_\lambda = \{1^m \mid \exists i (1 \leq i \leq s)$ such that $m \bmod p_i^{k_i} = 0\}$ can be accepted by a 1nfa in simple normal form with s cycles of lengths $p_1^{k_1}, p_2^{k_2}, \ldots, p_s^{k_s}$. Now, it is enough to apply the lower bound given in Theorem 3.1. □

5. Conclusion and open problems

We conclude the paper by briefly recollecting simulation results in the cyclic case, and by discussing their optimality. If we should sum up in a diagram the simulation costs of unary automata that accept cyclic languages, then we would obtain a figure similar to Figure 1, with the only difference that the arc from 1nfa to 2dfa is now labeled with n (and not n^2). As can be seen in [3, 8], the optimality of the exponential simulations is witnessed by cyclic languages. Let us now turn to simulations of cost n. The optimality of 1nfa vs. 2dfa, and of 1nfa vs. 2nfa is contained in Theorem 4.3. For the other simulations, it should be observed that witness languages presented in [3, 8] are not cyclic. So, the question arises of whether such tightness can be guaranteed also by cyclic languages.

Let us start with the 1dfa vs. 1nfa simulation. For any integer $\lambda > 0$, consider the language $\mathscr{L}_\lambda = \{1^{\lambda k} \mid k \geq 0\}$. Such a language is accepted by a 1nfa in simple normal form consisting of an initial state and a single cycle of λ states. Now, observe that, if a 1nfa in simple normal form contains (after removing useless states) a cycle of length λ', then it accepts at least two strings 1^n, 1^m, with $m - n = \lambda'$. Hence, in the light of Theorem 3.1, one can easily conclude that, for any integer $\lambda > 0$, the minimum 1nfa accepting the language $\mathscr{L}_\lambda$ coincides with the minimum 1dfa accepting $\mathscr{L}_\lambda$, namely with the automaton consisting of a single cycle of λ states. In other words, we can conclude that the trivial simulation of 1dfa's by 1nfa's is optimal even for cyclic languages.

On the other hand, we currently do not know any example of λ–cyclic language, with $\lambda = p_1^{k_1} \cdot p_2^{k_2} \cdot \ldots \cdot p_s^{k_s}$, which requires, in order to be accepted on 2dfa's or 2nfa's, a number of states which is not polynomial in $p_1^{k_1} + p_2^{k_2} + \cdots + p_s^{k_s}$. In this sense, we are not able to conclude that the trivial simulations of 1dfa's by 2dfa's and by 2nfa's are optimal for cyclic languages. The best we can say, in the light of Theorem 3.1, is that these simulations are optimal if we consider λ–cyclic languages, with λ being a prime power. The existence of cyclic languages requiring on 2dfa's a number of states which is not polynomial in $p_1^{k_1} + p_2^{k_2} + \cdots + p_s^{k_s}$, is related to the main open question of Sakoda and Sipser concerning the cost of the optimal simulation of (unary) 2nfa's by 2dfa's. In fact, each properly λ–cyclic language $\mathscr{L}$ is accepted by a 2dfa with $\lambda = p_1^{k_1} \cdot p_2^{k_2} \cdot \ldots \cdot p_s^{k_s}$ states (namely, the minimum 1dfa), and, on the other hand, each 2nfa accepting it requires $p_1^{k_1} + p_2^{k_2} + \cdots + p_s^{k_s}$ states. The existence, for any λ, of a language for which these upper and lower bounds are optimal should implies that 2nfa's are not polynomially equivalent to 2dfa's.

References

[1] J.-C. Birget. State-complexity of finite-state devices, state compressibility and incompressibility. *Mathematical Systems Theory*, **26** (1993), 237–269.

[2] P. Berman and A. Lingas. On the complexity of regular languages in terms of finite automata. Technical Report **304**, Polish Academy of Sciences, 1977.

[3] M. Chrobak. Finite automata and unary languages. *Theoretical Computer Science*, **47** (1986), 149–158.

[4] J.E. Hopcroft and J.D. Ullman. *Introduction to automata theory, languages, and computation.* Addison–Wesley, Reading, MA, 1979.

[5] T. Jiang, E. McDowell, and B. Ravikumar. The structure and complexity of minimal nfa's over a unary alphabet. *Int. J. Found. Comp. Sc.*, **2** (1991), 163–182.

[6] E. Landau. Uber die maximalordung der permutationen gegebenen grades. *Archiv. der Math. und Phys.*, **3** (1903), 92–103.

[7] E. Landau. *Handbuch der lehre von der verteilung der primzahlen I.* Teubner, Leipzig, Berlin, 1909.

[8] C. Mereghetti and G. Pighizzini. Optimal simulations between unary automata. In *Symposium on Theoretical Aspects of Computer Science 1998 (STACS 98), Proceedings*, Lecture Notes in Computer Science **1373**, Springer Verlag, 1998, 139–149. An extended version is available at: `http://www.dsi.unimi.it/~pighizzi/pubbl/unary`

[9] M.O. Rabin and D. Scott. Finite automata and their decision problems. *IBM J. Res. Develop*, **3** (1959), 198–200. Also in: E.F. Moore ed., *Sequential machines: selected papers*, Addison–Wesley, Reading, MA, 1964, 63–91.

[10] M. Sipser. Lower bounds on the size of sweeping automata. *J. Computer and System Science*, **21** (1980), 195–202.

[11] W.J. Sakoda and M. Sipser. Nondeterminism and the size of two–way finite automata. In *Proc. 10th ACM Symposium on Theory of Computing*, 1978, 275–286.