Counting with Automata

Orna Kupferman
Hebrew University*

Amnon Ta-Shma†
ICSI Berkeley

Moshe Y. Vardi
Rice University‡

July 19, 1999

Abstract

One of the natural examples of the power of concurrency is the ability to count to n with $\log n$ concurrent processes. In this work we study the cost of counting within weaker types of concurrency, where no cooperation takes place between the concurrent processes, except when time comes to decide whether the input should be accepted. We first study the number of states required for deterministic, nondeterministic, universal, and alternating automata to count to n, namely to accept the unary language $\{1^n\}$. We then study settings in which the task of counting should be accomplished before the whole input is read (e.g., in the language $\Sigma^{n-1} \cdot 0 \cdot \Sigma^*$). We show that then, the weaker types of concurrency are not helpful.

1 Introduction

Classical models of computations, such as Turing machines and automata, have been enriched with existential and universal branching to capture concurrency. Unlike the models used in the study of real concurrent systems, in these types of branching no cooperation takes place between the spawned precesses, except when time comes to decide whether the input should be accepted. It turned out that this weak cooperation is sufficient to make nondeterministic and universal automata exponentially more succinct than deterministic automata, and to make their combination, namely alternating automata, doubly exponentially more succinct than deterministic automata [CKS81]. Nevertheless, as studied in [DH94], enriching automata with real concurrency, where the spawned processes can cooperate all along the computation, results in even more succinct automata. In particular, concurrent alternating automata are triply exponentially more succinct than deterministic automata.

One of the natural examples of the power of concurrency is the ability to count to n with $\log n$ concurrent processes, with the i'th processor being responsible to the value of the i'th bit

*Address: Institute of Computer Science, Jerusalem 91904, Israel. Email: orna@cs.huji.ac.il

†Address: International Computer Science Institute, 1947 Center St., Berkeley, CA 94704. Email: amnon@icsi.berkeley.edu

‡Address: Department of Computer Science, Houston, TX 77251-1892, U.S.A. Email: vardi@cs.rice.edu.

in the boolean representation of n. In this work we study the cost of counting within the other types of concurrency.

We first study the number of states required for an automaton to count to n, namely to accept the unary language $\{1^n\}$. It is easy to see that nondeterministic automata need $O(n)$ states to count to n. For other types of automata, the problem was stated in [MF71], and was studied in [Bir93, Bir96] for alternating and two-way automata. Here we complete the picture with results about universal automata. We show that nondeterminism, universality, and alternation induce three different counting costs: like deterministic automata, nondeterministic automata need $\Theta(n)$ states to count to n. Then, universal automata need only $\Theta(\sqrt{n})$ states, and finally, alternating automata can count to n with $\Theta(\log n)$ states. These bounds are tight. Some of our results are based on known succinctness relations between the various types of automata with respect to unary languages [Lei81, Cho86, MP98], yet not all lower bounds that hold for general unary languages hold for $\{1^n\}$. For example, unlike the case for general unary languages, where nondeterministic automata are more succinct than one-way automata, here, nondeterminism does not help with counting.

In many applications, the only task of concurrency is efficient counting. The above results show that weaker types of concurrency are sufficient for efficient counting, thus they question the necessity of full concurrency in such applications. To approve the necessity of full concurrency, we also discuss "purposeful counting", where the task of counting should be accomplished before the whole input is read. Formally, we say that a given type of automata purposefully counts to n with $f(n)$ states iff for every language L that is accepted by an automaton of this type with k states, the language $(0+1)^{n-1} \cdot 0 \cdot L$ is accepted by an automaton of this type with $k + f(n)$ states. We show that all types of bounded concurrency cannot purposefully count efficiently, thus even alternating automata purposefully count to n with $\Theta(n)$ (rather than $\Theta(\log n)$) states.

2 Automata on finite words

Given an alphabet Σ, a *word* over Σ is a finite sequence $w = \sigma_1 \cdots \sigma_n$ of letters in Σ. We denote by w^l the suffix $\sigma_l \cdot \sigma_{l+1} \cdot \sigma_{l+2} \cdots$ of w. A *nondeterministic automaton on finite words* is $\mathcal{A} = \langle \Sigma, Q, q_{in}, M, F \rangle$, where Σ is the input alphabet, Q is a finite set of states, $M : Q \times \Sigma \to 2^Q$ is a transition function, $q_{in} \in Q$ is an initial state, and $F \subseteq Q$ is an acceptance condition. Intuitively, $M(q, \sigma)$ is the set of states that $\mathcal{A}$ can move into when it is in state q and it reads the letter σ. Since $\mathcal{A}$ may have several initial states and its transition function may specify many possible transitions for each state and letter, $\mathcal{A}$ is not *deterministic*. A *run* of $\mathcal{A}$ on $w = \sigma_1 \cdots \sigma_n$ is a function $r : \{0, \ldots n\} \to Q$ where $r(0) = q_{in}$ (i.e., the run starts in the initial state) and for every $l \geq 1$, we have $r(l) \in M(r(l-1), \sigma_l)$ (i.e., the run obeys the transition function). The run is *accepting* iff $r(n) \in F$ (i.e., the run terminates in one of the accepting states). The *language* $\mathcal{L}(\mathcal{A})$ of $\mathcal{A}$ is the set of words $w \in \Sigma^*$ for which there exists an accepting run of $\mathcal{A}$ on w. The automaton $\mathcal{A}$ is *unary* if $|\Sigma| = 1$.

Since $\mathcal{A}$ is not deterministic, it may have many runs on w. There are two dual ways in which we can refer to the many runs. When $\mathcal{A}$ is a nondeterministic automaton as above, it accepts

an input word w iff there exists an accepting run of $\mathcal{A}$ on w. Dually, when $\mathcal{A}$ is a *universal* automaton, it accepts an input word w iff all the runs of $\mathcal{A}$ on w are accepting. *Alternating* automata [BL80, CKS81] combine both existential and universal branching choices.

An alternating automaton is $\mathcal{A} = \langle \Sigma, Q, q_{in}, M, F, \rho \rangle$, where Σ, Q, q_{in}, M, and F are as in nondeterministic automata, and $\rho : Q \to \{\exists, \forall\}$ is a *classification function* that classifies each state q as either existential (in case $\rho(q) = \exists$), or universal (in case $\rho(q) = \forall$). The intuition is that the automaton accepts a suffix w^l of w from an existential state q if it accepts w^{l+1} from some state in $M(q, \sigma_l)$, and it accepts w^l from a universal state q if it accepts w^{l+1} from all states in $M(q, \sigma_l)$.

Formally, a run of the alternating automaton $\mathcal{A}$ on w is a function $r : \{0, \ldots n\} \to 2^Q$ where $r(0) = \{q_{in}\}$ and for every $l \geq 1$ and $q \in r(l)$, either $\rho(q) = \exists$, in which case $M(q, \sigma_l) \cap r(l+1) \neq \emptyset$, or $\rho(q) = \forall$, in which case $M(q, \sigma_l) \subseteq r(l+1)$. Thus, the run chooses a successor of existential states and proceeds with all successors of universal states. The run is accepting iff $r(n) \subseteq F$ (i.e., all "copies" of the run terminates in accepting states).

Deterministic, nondeterministic, and universal automata can be viewed as special cases of alternating automata. Formally, an alternating automaton is deterministic if for all q and σ, the set $M(q, \sigma)$ is either empty or a singleton. An automaton is nondeterministic if all states are existential, and it is universal if all states are universal. The classification of states to existential and universal makes the *complementation* problem for alternating automata as easy as the problem of complementing deterministic automaton. For a classification function ρ, let $\tilde{\rho}$ be dual to ρ, thus, for all $q \in Q$, we have that $\tilde{\rho}(q) = \forall$ iff $\rho(q) = \exists$ and $\tilde{\rho}(q) = \forall$ iff $\rho(q) = \exists$. Given $\mathcal{A} = \langle \Sigma, Q, q_{in}, M, F, \rho \rangle$, it is easy to see that the automaton $\tilde{\mathcal{A}} = \langle \Sigma, Q, q_{in}, M, Q \setminus F, \tilde{\rho} \rangle$ accept exactly all words rejected by $\mathcal{A}$.

3 Counting with Automata

For $n \in \mathbb{N}$, we define the language $C_n = \{1^n\}$ over the alphabet $\{1\}$. We say that an automaton $\mathcal{A}$ *counts to* n iff $\mathcal{A}$ recognizes C_n. We say that a type of automata *counts to* n *with* $f(n)$ *states* if there is an automaton of this type that accepts C_n with $f(n)$ states. Since we deal with unary languages, then if $\mathcal{A}$ accepts 1^k for some $k \geq 0$, we say that $\mathcal{A}$ accepts k and we write $k \in \mathcal{A}$.

3.1 Deterministic and nondeterministic automata

Theorem 3.1 *For every $n \geq 0$, deterministic and nondeterministic automata count to n with $n + 1$ states.*

Proof: For the upper bound, consider the deterministic automaton $\mathcal{A} = \langle \{1\}, \{0, \ldots n\}, 0, M, \{n\} \rangle$ where $M(i, 1) = \{i + 1\}$ for $i \leq n$ and $M(n, 1) = \emptyset$. It is easy to see that $\mathcal{A}$ recognizes C_n. For the lower bound, assume by way of contradiction that there exists a nondeterministic automaton $\mathcal{A}_n$ with less than $n + 1$ states that recognizes C_n. Since an accepting run of $\mathcal{A}_n$ on 1^n must visit at least one state more than once, the transition function of $\mathcal{A}_n$ contains a cycle that enables the acceptance of more words, not in C_n, and we reach a contradiction. □

3.2 Alternating automata

It is proved in [Lei81] that every unary deterministic automaton with n states has an equivalent alternating automaton with $\lceil \log n \rceil$ states. Hence, Theorem 3.1 immediately implies the following theorem.

Theorem 3.2 *For every $n \geq 0$, alternating automata count to n with $\lceil \log n \rceil$ states.*

We say that an alternating automaton is a *one-switch* alternating automaton if it cannot move from existential states to universal states. Thus, a one-switch alternating automaton can change its "branching mode" at most once, from a universal mode to an existential mode. The following theorem is proven in [Bir96] and we give it here with its proof, which we would use later.

Theorem 3.3 [Bir96] *For every $n \geq 0$, one-switch alternating automata count to n with $O(\log^2 n \log\log n)$ states.*

Proof: Given n, let $p_1, \ldots, p_k$ be the first k prime numbers such that $n \leq p_1 \cdot p_2 \cdots p_k$. By the Chinese Remainder Theorem, there is a tuple $\langle a_1, \ldots, a_k \rangle$ such that for all $1 \leq i \leq k$, we have $n = a_i \bmod p_i$, and n is the unique number smaller than $p_1 \cdot p_2 \cdots p_k$ with the above property. That is, for all $n \neq m \leq p_1 \cdot p_2 \cdots p_k$, there exists $1 \leq i \leq k$ such that $m \neq a_i \bmod p_i$. An alternating automaton $\mathcal{A}$ that recognizes C_n has, for each $1 \leq i \leq k$, two cycles each of size p_i. The first, i-good, cycle, accepts all words of length $l = a_i \bmod p_i$. The second, i-bad, cycle, accepts all words of length $l \neq a_i \bmod p_i$. Given an input word w, the alternating automaton checks that for all $1 \leq i \leq k$, we have $|w| = a_i \bmod p_i$, and that no proper suffix of w has this property. For that, $\mathcal{A}$ uses its universality to make sure that w is accepted by all the i-good cycles, and to make sure that all the proper suffixes of w are accepted by some, nondeterministically chosen, i-bad cycle. Since $p_i = O(i \log i)$ and $k \leq \log n$, we have that $p_i \leq \log n \log\log n$, and we are done. □

The automaton $\mathcal{A}$ described in the proof is "almost universal". Indeed, using the Chinese Reminder Theorem, we can easily define a universal automaton with *polylog*(n) states that accepts exactly all numbers k for which $k = a_i \bmod p_i$ for all $1 \leq i \leq k$. Then, nondeterminism is used only to "mask" all k's except the smallest one that satisfy the above. Can we do define such a mask without nondeterminism? In the next section we show that we can, yet while such a nondeterministic mask requires only *polylog*(n) states, a universal one requires $\Theta(\sqrt{n})$ states.

3.3 Universal automata

The *greatest common divisor* of $a_1, \ldots, a_k \in \mathbb{N}$, denoted $gcd(a_1, \ldots, a_k)$, is the largest integer d dividing all a_i's. Two integers x and y are *relatively prime* if $gcd(x, y) = 1$. For two integers x and y, with $x \leq y$, we define the nondeterministic automaton $\mathcal{B}_{x,y} = \langle \{1\}, \{0, \ldots, y-1\}, 0, M, \{0\} \rangle$,

where $M(0) = \{1, y-x+1\}$, and for all $i > 0$, we have $M(i) = i+1 \bmod y$. Thus, $\mathcal{B}_{x,y}$ has two loops, one of length x and the other of length y, both connected at the state 0. Therefore, $\mathcal{B}_{x,y}$ accepts exactly these numbers of the form $ax + by$ for some $a, b \in \mathbb{N}$. It is known (see [Cho86]) that if x and y are relatively prime numbers, then the greatest c such that $ax + by = c$ has no solution in natural numbers is $(x-1)(y-1) - 1$. This immediately implies the following property of $\mathcal{B}_{x,y}$.

Lemma 3.4 *Let x and y be two relatively prime numbers, and let $n' = (x-1)(y-1) - 1$. The automaton $\mathcal{B}_{x,y}$ rejects n' and accepts all integers $k > n'$.*

For example, if $x = 10$ and $y = 11$, then $n' = 89$. The nondeterministic automaton $\mathcal{B}_{10,11}$ has 11 states. It rejects 89 and accepts all integers $i > 90$. Lemma 3.4 provides the mask we were looking for. We can now use $\mathcal{B}_{x,y}$ and Lemma 3.4 to count with universal automata.

Theorem 3.5 *For every $n \geq 0$, universal automata count to n with $\Theta(\sqrt{n})$ states.*

Proof: We start with the upper bound. We describe a dual construction, namely a nondeterministic automaton $\mathcal{A}$ such that $\mathcal{A}$ accepts exactly all integers $k \neq n$. Then, the automaton $\tilde{\mathcal{A}}$ counts to n. Given n, we define $\mathcal{A}$ as the union of two nondeterministic automata, $\mathcal{A}_n$ and $\mathcal{B}_n$. The automaton $\mathcal{A}_n$ rejects n, accepts all $k < n$, and we do not care about its behavior with respect to integers $k > n$. The automaton $\mathcal{B}_n$ rejects n, accepts all $k > n$, and we do not care about its behavior with respect to integers $k < n$. Since $\mathcal{A}$ accepts iff either $\mathcal{A}_n$ or $\mathcal{B}_n$ accept, $\mathcal{A}$ accepts all integers $k \neq n$. The automaton $\mathcal{A}_n$ has $O(\log^2 n \cdot \log\log n)$ states and the automaton $\mathcal{B}_n$ has $O(\sqrt{n})$ states. Then, $\tilde{\mathcal{A}}$ is a universal automaton that counts to n with $O(\sqrt{n})$ states.

The automaton $\mathcal{A}_n$ is similar to the automaton described in the proof of Theorem 3.2. Let $p_1, \ldots, p_k$ be the first k prime numbers such that $p_1 \cdot p_2 \cdots p_k \geq n$. For each $1 \leq j \leq k$, the automaton $\mathcal{A}_n$ contains a cycle that accepts all integers of length $l \neq a_j \bmod p_j$. Thus, $\mathcal{A}_n$ accepts $i \geq 0$ iff there is $1 \leq j \leq k$ such that $i \neq a_j \bmod p_j$. By the Chinese reminder theorem, $\mathcal{A}_n$ rejects n, accepts all $i < n$, and its behavior with respect to integers $i > n$ is arbitrary. Since $p_i = O(i \log i)$ and $k \leq \log n$, we have that $p_i \leq \log n \log\log n$, and the automaton $\mathcal{A}_n$ has $O(\log^2 n \log\log n)$ states.

The automaton $\mathcal{B}_n$ is based on an automaton $\mathcal{B}_{x,y}$ for appropriate x and y. Given $n \in \mathbb{N}$, let $x = \lfloor \sqrt{n} \rfloor$ and $y = x + 1$. Since no prime can divide both x and $x+1$, the numbers x and y are relatively prime. Therefore, by Lemma 3.4, the nondeterministic automaton $\mathcal{B}_{x,y}$ rejects $n' = (x-1)(y-1) - 1$ and accepts all $i > n'$ with $y \leq \sqrt{n} + 1$ states. The automaton $\mathcal{B}_n$ is now obtained by preceding the initial state of $\mathcal{B}_{x,y}$ with a chain of $n - n'$ states. Thus, B_n reads the first $n - n'$ letters in the input and then runs $\mathcal{B}_{x,y}$. As $\sqrt{n} - 1 < x \leq \sqrt{n}$, a simple calculation shows that $0 \leq n - n' < 3\sqrt{n}$, implying that $\mathcal{B}_n$ has at most $4\sqrt{n}$ states.

It is left to prove a lower bound. In [Bir96], it is shown that two-way deterministic automata (automata in which the reading head can move both forward or backward over the input) need $n + 2$ states in order to count to n. Since two-way deterministic automata are easy to

complement, this implies that two-way deterministic automata need $n+2$ states in order to accept exactly all integers $k \neq n$. The lower bound then follows from fact that every n-state nondeterministic automaton over a unary language can be translated to an $O(n^2)$-state two-way deterministic automaton [Cho86]. □

Remark 3.6 While for general unary languages, universal two-way automata are more succinct than one-way automata [Cho86], the ability to go back and forth the input is not helpful for universal automata in the context of counting. To see that, we prove that a two-way nondeterministic automaton needs $O(\sqrt{n})$ states to recognize $\{1^k : k \neq n\}$. In [MP98], it is proved that if a unary language is accepted by a two-way nondeterministic automaton $\mathcal{A}$ with x states, then there is a constant $l < O(e^{\sqrt{x \ln x}})$ such that for every integer $k > 8x^2$, we have that $1^k \in \mathcal{L}(\mathcal{A})$ iff $1^{k+l} \in \mathcal{L}(\mathcal{A})$. So if there is an automaton with $x < \sqrt{\frac{n}{8}}$ states for $\{1^k : k \neq n\}$, its behavior on 1^n coincides with its behavior on numbers larger than n, which is wrong. □

4 Purposeful Counting with Automata

For $n \in \mathbb{N}$ and a regular language L, we define the language $L_n^0 = (0+1)^{n-1} \cdot 0 \cdot L$. Note that all the words in L_n^0 have 0 in their n'th position. We say that a type of automata *purposefully counts to n with $f(n)$ states* iff for every language L that is accepted by an automaton of this type with k states, the language L_n^0 is accepted by an automaton of this type with $k+f(n)$ states. Intuitively, unlike an automaton that merely counts to n, an automaton that purposefully counts to n is able to use its counting information in order to take decisions other than acceptance during the run.

Theorem 4.1 *For every n, deterministic, nondeterministic, universal, and alternating automata purposefully count to n with $\Theta(n)$ states.*

Proof: For the upper bound, given n and L, let $\mathcal{B} = \langle \Sigma, Q, q_{in}, M, F, \rho \rangle$ be the automaton for L. Consider the automaton $\mathcal{A} = \langle \Sigma \cup \{0,1\}, Q \cup \{0, \ldots n\}, 0, M', F, \rho' \rangle$, where for $q \in Q$ and $\sigma \in \Sigma$ we have $M'(q,\sigma) = M(q,\sigma)$, and for the new states $\{0, \ldots n\}$, we have $M'(i,\sigma) = i+1$ for $i < n$ and $\sigma \in \{0,1\}$, $M(n,0) = q_{in}$, and $M(n,1) = \emptyset$. Also, ρ' agrees with ρ about the classification of states in Q and maps all states in $\{0, \ldots n\}$ to $\exists$. It is easy to see that $\mathcal{B}$ recognizes L_n^0.

For the lower bound, we first define, for $n \in \mathbb{N}$, the language $P_n^0 = (0+1)^{n-1} \cdot 0 \cdot (0+1)^*$ over the alphabet $\{0,1\}$. Thus, P_n^0 is L_n^0 for $L = (0+1)^*$. Consider the regular language

$$\mathcal{L}_n = \{\{0,1,\#\}^* \cdot \# \cdot w \cdot \# \cdot \{0,1,\#\}^* \cdot \$ \cdot w \ : w \in \{0,1\}^n\}.$$

A word τ is in $\mathcal{L}_n$ iff its suffix of length n that comes after the single \$ in τ appears somewhere between #'s before the \$. By [CKS81], the smallest deterministic automaton that recognizes $\mathcal{L}_n$ has at least 2^{2^n} states (reaching the \$, the automaton should remember the possible set

of words in $\{0,1\}^n$ that have appeared before). We show that if P_n^0 can be recognized by an alternating automaton with m states, then $\mathcal{L}_n$ can be recognized by an alternating automaton with $O(m+\log n)$ states, implying it also can be recognized by a deterministic automaton with only $2^{2^{O(m+\log n)}}$ states [CS76]. Hence, $m = \Theta(n)$.

Assume that P_n^0 can be recognized by an alternating automaton $\mathcal{A}_0$ with m states. We can modify $\mathcal{A}_0$ to an automaton $\mathcal{A}_0'$ that recognizes the language

$$R_n^0 = \bigcup_{1 \leq i \leq n-1} (0+1)^i \cdot \# \cdot (0+1+\#)^* \cdot \$ \cdot (0+1)^{n-(i+1)} \cdot 0 \cdot (0+1)^*.$$

Intuitively, each word in R_n^0 is obtained from a word in P_n^0 by inserting a word in $\#\cdot(0+1+\#)^*\cdot\$$ somewhere among the first $n-1$ letters. Accordingly, the automaton $\mathcal{A}_0'$ starts as $\mathcal{A}_0$ and it moves to a suspension state (in all its copies) when it encounters the first # in the input. From the suspension state it goes back to the state in which it has been suspended when it encounters the first \$. This involves a duplication of $\mathcal{A}_0$'s state space and a fixed number of new states. We can similarly define an analogous automaton $\mathcal{A}_1'$ that recognizes the language

$$R_n^1 = \bigcup_{1 \leq i \leq n-1} (0+1)^i \cdot \# \cdot (0+1+\#)^* \cdot \$ \cdot (0+1)^{n-(i+1)} \cdot 1 \cdot (0+1)^*.$$

Now, we can use the automata $\mathcal{A}_0'$ and $\mathcal{A}_1'$ in order to construct an alternating automaton with $O(m+\log n)$ states for $\mathcal{L}_n$. The idea is similar to the construction of an alternating automaton with concurrency for $\mathcal{L}_n$ in [DH94]. The automaton traverses the input word and guesses a position (right after a # letter) in which w starts. Then, the automaton needs to check that for all $1 \leq i \leq n$, the i'th letter in w agrees with the i'th letter after the \$. To do this, as long as the next # does not appear in the input, the automaton universally branch in each state to either the initial state of $\mathcal{A}_0'$, in case the letter in the input is 0, or to the initial state of $\mathcal{A}_1'$, in case the letter in the input is 1. Finally, when the automaton encounters the first \$ in the input, it uses $\log n$ states to counts to n in order to ensure that the guessed w is of length n. □

Theorem 4.1 shows that the disability of the spawned copies of the automaton to collaborate during the processing of the input word is crucial.

Remark 4.2 In our definition of purposeful counting, the termination of the task of counting may not coincide with the end of the input, which disables a collaboration of the spawned copies of the automaton. On the other hand, the start of the task of counting does coincide with the beginning of the input. Consequently, it is not hard to see that for every n, two-way alternating automata can purposefully count to n with $\Theta(\log n)$ states. Indeed, in order to recognize L_n^0, a two-way automaton can guess the 0 in position n and then universally branches into two copies. One counts backwards to n with $\log n$ states (as in Theorem 3.2), and one follows the automaton for L. Nevertheless, a definition of purposeful counting in which both the start and the termination of the task of counting do not necessarily coincide with the beginning and end of the input (say by defining $L_n^0 = L \cdot 0 \cdot (0+1)^{n-1} \cdot 0 \cdot L$) would prevent alternating two-way automata from efficient purposeful counting. □

References

[Bir93] J.C. Birget. State-complexity of finite-state devices, state compressibility and incompressibility. *Mathematical Systems Theory*, 26(3):237–269, 1993.

[Bir96] J.C. Birget. Two-way automata and length-preserving homomorphisms. *Mathematical Systems Theory*, 29(3):191–226, 1996.

[BL80] J.A. Brzozowski and E. Leiss. Finite automata and sequential networks. *Theoretical Computer Science*, 10:19–35, 1980.

[Cho86] M. Chorbak. Finite automata and unary languages. *Theoretical Computer Science*, 47:149–158, 1986.

[CKS81] A.K. Chandra, D.C. Kozen, and L.J. Stockmeyer. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133, January 1981.

[CS76] A.K. Chandra and L.J. Stockmeyer. Alternation. In *Proc. 17th IEEE Symposium on Foundations of Computer Science*, pages 98–108, New York, 1976.

[DH94] D. Drusinsky and D. Harel. On the power of bounded concurrency I: Finite automata. *Journal of the ACM*, 41(3):517–539, 1994.

[Lei81] E. Leiss. Succinct representation of regular languages by boolean automata. *Theoretical computer science*, 13:323–330, 1981.

[MF71] A.R. Meyer and M.J. Fischer. Economy of description by automata, grammars, and formal systems. In *Proc. 12th IEEE Symp. on Switching and Automata Theory*, pages 188–191, 1971.

[MP98] C. Mereghetti and G. Pighizzini. Optimal simulations between unary automata. In *Proc. 15th Symp. on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science, 1998.