Counting from the End Needs Exponentially Many States, Counting from the Start Only Linearly Many

— a nondeterministic machine reading "the n-th symbol from the end is 1" has $n+1$ states, but made deterministic it needs exactly 2^n, which minimisation does not reduce — the separator is whether the machine must commit to one reading without looking ahead or may branch on guesses — [Paper 765]

Yuuki YAMAGISHI (山岸 勇輝)
Yamagishi Complexity Torus Lab, Japan
September 2026

— *From thinking to syncing.* —

Abstract

Languages written as regular expressions can be read by machines with finite memory. But how much memory they need differs greatly, even for expressions of the same length.Machine states are counted for a language counted from the end and one counted from the start.No new theorem or law is claimed. First, nondeterministic machines need only linearly many states for either——"n-th from the end is 1" takes $n+1$ states (Section 2). **Second, made deterministic, counting from the end reaches exactly 2^n states——subset construction's reachable states counted for $n=1$ to 12 all equal 2^n** (Section 2). **Third, minimisation does not reduce them——for $n=1$ to 10 the minimal deterministic machine also has 2^n states** (Section 2). **Fourth, and this is the core. Counting from the start, even a deterministic machine needs only linearly many——$n+2$ states at minimum; at $n=10$, 1024 against 12, 85.333333 times** (Section 3). **Fifth, both machines read exactly their languages——checked at $n=3$ on all 8191 words up to length 12** (Section 3). **Sixth, the separator is whether the machine must commit to one reading without looking ahead or may branch on guesses——"regular expressions are read with finite memory" does not say how much until the counting direction and determinism are stated** (Section 4). **Keywords**: regular expressions, finite automata, subset construction, nondeterminism, state explosion, counting direction.

Scope of this paper (scope note): **No new theorem or law is claimed——making a nondeterministic finite automaton deterministic by subset construction, that its states can grow exponentially, that the "n-th from the end" language is the standard example needing exponentially many states, and state minimisation are all known (Rabin and Scott 1959, Hopcroft et al. 2006).Implementation is not treated——matching implementations, backtracking and speed are not treated.Two symbols——only languages over the two symbols 0 and 1 are considered.No upper bound is proved——general bounds on state explosion are only cited; states of this family are merely counted.Ranges are limited——minimisation was carried to $n = 10$ and reachable-state counting to $n = 12$.Relation to earlier papers**: **Paper 296 showed that the hierarchy of languages is set by the type of memory, not its amount——its scope note stated that it used one pigeonhole argument and three comparisons of effort, and only cited the Myhill-Nerode theorem. There type decided the hierarchy. Here, within the same type, regular languages, the amount of memory splits into 2^n and $n + 2$, and what splits it is counting direction and determinism. Differences of type, and differences of amount within a type.**What is added is **counting subset construction's reachable states to $n = 12$ and matching 2^n**, **confirming to $n = 10$ that minimisation does not reduce them**, **setting the minimum $n + 2$ for counting from the start beside it, a factor of $\mathbf{85.333333}$ at $n = 10$**, **checking both machines' languages exhaustively**, and **placing the separator on committing to one reading or branching**.

1. Notation and the Question

Setting: **over words of the symbols 0 and 1, two languages are written as regular expressions of similar length.**

$$L_{\text{end}}(n) = (0 \mid 1)^* \, 1 \, (0 \mid 1)^{n-1}, \qquad L_{\text{start}}(n) = (0 \mid 1)^{n-1} \, 1 \, (0 \mid 1)^*$$

$L_{\text{end}}(n)$ is the set of words whose n-th symbol from the end is 1; $L_{\text{start}}(n)$, whose n-th from the start is 1. A nondeterministic machine may branch on guesses as it reads; a deterministic one commits to a single state after each symbol.

Question: **do regular expressions of the same length need machines of about the same memory?**

2. Counting from the End: 2^n States

Computation 2.1: **the nondeterministic machine for $L_{\text{end}}(n)$ ($n + 1$ states) was made deterministic by subset construction, its reachable states counted, and then minimised.**

n	nondeterministic states	reachable after subset construction	minimal deterministic
1	2	2	2
3	4	8	8
5	6	32	32
8	9	256	256
10	11	1024	1024
12	13	4096	——

> **Observation 2.2**
>
> made deterministic, counting from the end reaches exactly 2^n states.All cases from $n = 1$ to 12 matched—— a deterministic machine must remember every arrangement of the last n symbols.

> **Observation 2.3**
>
> minimisation does not reduce them.For $n = 1$ to 10 the minimal deterministic machine also has 2^n states ——this is not waste in the construction but memory the language itself requires.

3. Counting from the Start: Linear Suffices

Computation 3.1: a deterministic machine for $L_{\text{start}}(n)$ was built and minimised, and at $n = 3$ all words up to length 12 were fed to both machines.

n	from the end (minimal)	from the start (minimal)
1	2	3
3	8	5
5	32	7
10	1024	12

Observation 3.2

this is the core: counting from the start, even a deterministic machine needs only linearly many states.The minimum is $n+2$ (for $n=1$ to 12); at $n=10$, 12 states against 1024 from the end, a factor of 85.333333 ——counting from the start, the answer is fixed the moment the n-th symbol is read, and nothing more need be remembered.

Observation 3.3

both machines read exactly their languages.At $n=3$, on all 8191 words of length 0 to 12, acceptance matched the definition——the difference in states is not a construction error.

4. The Separator

Observation 4.1

the separator is whether the machine must commit to one reading without looking ahead or may branch on guesses.Allowed to branch, counting from the end takes $n+1$ states; committed, it takes 2^n. Counting from the start shows no such gap——"regular expressions are read with finite memory" does not say how much until the counting direction and determinism are stated.

Observation 4.2

beside Paper 296.Paper 296 showed that the hierarchy of languages is set by the type of memory, not its amount——here, within the same type of regular languages, the amount splits into 2^n and $n+2$. Type may set the hierarchy, while amount within a type moves exponentially with how one reads.

Observation 4.3

what is not said.Implementation, speed, alphabets of three or more symbols and general upper bounds are not treated——what is shown is only a count, in this family, of states splitting into exponential and linear.

5. Roll Call and Closing

1. **nondeterministic $n+1$ states** (Computation 2.1): **linear even from the end.**
2. **reachable 2^n** (Observation 2.2): **$n = 1$ to 12.**
3. **4096 at $n = 12$** (Computation 2.1): **the end of the count.**
4. **minimal also 2^n** (Observation 2.3): **memory of the language itself.**
5. **1024 at $n = 10$** (Computation 2.1): **deterministic from the end.**
6. **$n+2$ from the start** (Observation 3.2): **the core.**
7. **12 at $n = 10$** (Computation 3.1): **deterministic from the start.**
8. **85.333333 times** (Observation 3.2): **the difference of direction.**
9. **agreement on 8191 words** (Observation 3.3): **not a construction error.**
10. **commit or branch** (Observation 4.1): **the separator; Paper 296.**

Closing: **Counting from the end needs exponentially many states, counting from the start only linearly many.A nondeterministic machine reading "the n-th symbol from the end is 1" has $n+1$ states, but made deterministic it reaches exactly 2^n (for $n = 1$ to 12), and minimisation does not reduce them (for $n = 1$ to 10)——a deterministic machine must remember every arrangement of the last n symbols.Yet for "the n-th symbol from the start is 1" even a deterministic machine needs only $n+2$ states, so at $n = 10$ it is 1024 against 12, a factor of 85.333333——counting from the start, the answer is fixed the moment the n-th symbol is read.At $n = 3$ both machines accepted exactly their languages on all 8191 words up to length 12.The separator is whether the machine must commit to one reading without looking ahead or may branch on guesses——"regular expressions are read with finite memory" does not say how much until the counting direction and determinism are stated.Placed among the earlier papers——Paper 296 showed that type sets the hierarchy; this counts how, within one type, memory moves exponentially with the way of reading.To be honest——implementation, speed and general upper bounds are not treated; the states of this family were only counted.**

References

1. Hopcroft, J. E., Motwani, R., & Ullman, J. D. (2006). *Introduction to Automata Theory, Languages, and Computation* (3rd ed.). Addison-Wesley.
2. Rabin, M. O., & Scott, D. (1959). Finite automata and their decision problems. *IBM Journal of Research and Development*.
3. Yamagishi, Y. (2026). *The Hierarchy of Languages Is Set by the Type of Memory, Not Its Amount* [Paper 296]. Zenodo. 10.5281/zenodo.22171705

— From thinking to syncing. —

Yuuki YAMAGISHI (山岸 勇輝)

© 2026 Yamagishi Complexity Torus Lab, Japan · Licensed under Creative Commons Attribution 4.0 International (CC BY 4.0)