Windable Heads & Recognizing NL with Constant Randomness

M. Utkan Gezer

Department of Computer Engineering, Boğaziçi University
Bebek 34342, Istanbul, Turkey

LATA 2020
September 2021

Constant-Space, Constant-Randomness Verifiers with Arbitrarily Small Error

M. Utkan Gezer A.C. Cem Say

Department of Computer Engineering, Boğaziçi University
Bebek 34342, Istanbul, Turkey

LATA 2020
September 2021

Table of Contents

Motivation

- [Har72] proves $\text{NL} = \mathcal{L}(2\text{nfa}(k))$.
- [SY14] proves $\text{NL} = 1\text{IP}(\infty, \text{cons}, \text{cons})$.
- Naturally, $1\text{IP}_*(\infty, \text{cons}, \text{cons}) \subseteq 1\text{IP}(\infty, \text{cons}, \text{cons})$.
- Q: Which of the languages in $1\text{IP}(\infty, \text{cons}, \text{cons})$ can we verify with small error? (perhaps as a subset of NL or $\mathcal{L}(2\text{nfa}(k))$)

M. Utkan Gezer [utkan.gezer@boun.edu.tr], A.C. Cem Say

Motivation

- [Har72] proves $\text{NL} = \mathcal{L}(2\text{nfa}(k))$.
- [SY14] proves $\text{NL} = 1\text{IP}(\infty, \text{cons}, \text{cons})$.
- Naturally, $1\text{IP}_*(\infty, \text{cons}, \text{cons}) \subseteq 1\text{IP}(\infty, \text{cons}, \text{cons})$.
- Q: Which of the languages in $1\text{IP}(\infty, \text{cons}, \text{cons})$ can we verify with small error? (perhaps as a subset of NL or $\mathcal{L}(2\text{nfa}(k))$)

M. Utkan Gezer [utkan.gezer@boun.edu.tr], A.C. Cem Say

Results

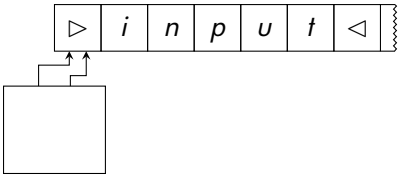
- $\mathcal{L}(2\text{nfa}(k), \text{linear-time}) \subseteq 1\text{IP}_*(\infty, \text{cons}, \text{cons})$
  - Safe heads imply membership in $1\text{IP}_*(\infty, \text{cons}, \text{cons})$.
  - Safe heads imply linear runtime, and vice versa.
- $\mathcal{L}(2\text{nfa}(k), \text{linear-time}) \subseteq \text{NTISP}(n^2/\log(n), \log n)$

M. Utkan Gezer [utkan.gezer@boun.edu.tr], A.C. Cem Say Department of Computer Engineering, Boğaziçi University, Istanbul, Turkey

Table of Contents

Multi-head finite automata

A *2nfa*(k) is a nondeterministic finite automaton with k read-only heads on a tape with $\triangleright w \triangleleft$ written on it, where w is the input. An example 2nfa(2):

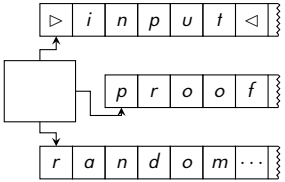


Guesses its way to acceptance, if there's any.

$\mathcal{L}(2\text{nfa}(k))$ class recognized by finite automata with any number of heads

2pfa verifiers

A *2pfa verifier* is a finite automaton modified to have one-way access to the most convincing purported proof of membership for its input and a sequence of random bits.



$1\text{IP}(\infty, \text{cons}, \text{cons})$ limited work space and random bits, and < 0.5 "error"

$1\text{IP}_*(\infty, \text{cons}, \text{cons})$ arbitrarily reducible error

M. Utkan Gezer [utkan.gezer@boun.edu.tr], A.C. Cem Say

Error of a 2pfa verifier

- We expect our 2pfa verifiers to accept members and reject nonmembers.
- The (strong) error of a 2pfa V is the maximum rate that it fails given any input and the most convincing certificates.

Table of Contents

Multiple heads $\equiv$ logarithmic space

Lemma ([Har72])

Multiple heads can simulate a logarithmic space, and vice versa.

$$\mathcal{L}(2nfa(k)) = \text{NL}$$

Necessary number of heads

Lemma ([Mon80])

More heads are strictly more powerful. Thus, for any $k > 0$:

$$\mathcal{L}(2nfa(k)) \subsetneq \mathcal{L}(2nfa(k+1))$$

Consequently, there is a minimum number of heads to recognize a language.

Bounded runtime

Lemma (Multi-head finite automata with bounded runtime)

Runtime can be bounded by doubling the number of heads. More precisely, for $k > 0$:

$$\mathcal{L}(2nfa(k)) \subseteq \mathcal{L}(2nfa(2k), n^k)$$

We qualify machines with bounded runtime as (always) halting.

M. Utkan Gezer [utkan.gezer@boun.edu.tr], A.C. Cem Say

Necessary number of heads for a bounded runtime

Corollary

There is a minimum number of heads to recognize a language A in bounded runtime.

We denote that with k_A.

Simulating many heads with one and some coins

The attempt

- Let M be a halting 2nfa(k) recognizing the language A.
- The 2pfa verifier V recognizes A by simulating M:
  - The certificate describes an accepting execution of M with
    - head readings,
    - nondeterministic choices.
  - V simulates M solely by the certificate.
  - Traces a random head of M and verifies its readings.
  - Repeats for m rounds.
- V accepts if all rounds pass.

Idea from [SY14].

M. Utkan Gezer [utkan.gezer@boun.edu.tr], A.C. Cem Say

Simulating many heads with one and some coins

The analysis

- Member certificates are always accepted.
- Nonmember certificates lie towards
  - acceptance and get caught with probability $1/k$ each round—*OK,*
  - looping (!) and get caught with probability $1/k$ overall—*not OK.*
- M being windable in its single-headed simulation defeats the algorithm.

M. Utkan Gezer [utkan.gezer@boun.edu.tr], A.C. Cem Say

Simulating many heads with one and some coins

The analysis

- Member certificates are always accepted.
- Nonmember certificates lie towards
  - acceptance and get caught with probability $1/k$ each round—*OK,*
  - looping (!) and get caught with probability $1/k$ overall—*not OK.*
- M being windable in its single-headed simulation defeats the algorithm.

Simulating many heads with one and some coins

Safe heads and the fix

- We can recover those M with a *safe head*—heads that are robuts against looping in single-headed simulations. We just bias towards choosing it.
- Probabilities of V...

p_s	choosing the safe head
p_{ll}	choosing the least likely head
$1 - p_s$	looping on the first round (it's less every other round)
$(1 - p_{ll})^m$	falsely accepting

- Increasing p_s and m while ensuring $p_{ll} > 0$ reduces the error arbitrarily.

M. Utkan Gezer [utkan.gezer@boun.edu.tr], A.C. Cem Say

Simulating many heads with one and some coins

Safe heads and linear time

- Let M be a 2nfa(k) with a safe head.
  - Single-headed simulations run for $\leq |Q| \cdot (|w| + 2)$ steps, where Q is the set of states.
  - Full simulation is more restricted, and at least as limited.
- Let M be a 2nfa(k) running in linear time.
  - M runs for $\leq c \cdot |w|$ steps for some constant c.
  - Add a timer head moving once every c^{th} step and faults if it reaches the end.
  - The new head is safe.

M. Utkan Gezer [utkan.gezer@boun.edu.tr], A.C. Cem Say

Simulating many heads with one and some coins

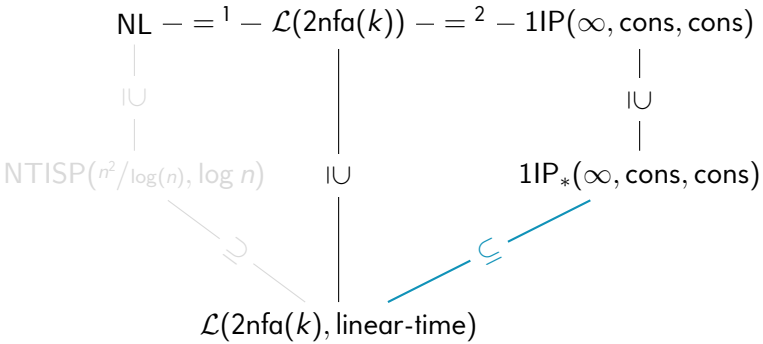
Safe heads

Being a safe head of a 2nfa(k) is a decidable property [GS21].

M. Utkan Gezer [utkan.gezer@boun.edu.tr], A.C. Cem Say

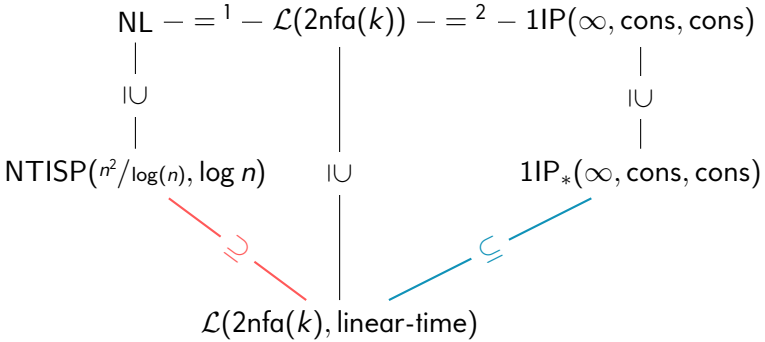
Table of Contents

Overview



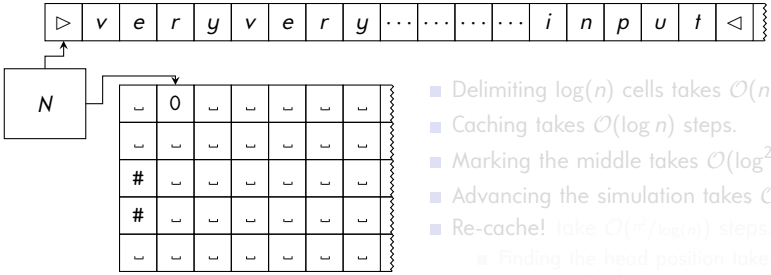
[1] [Har72]
[2] [SY14]

Overview



[1] [Har72]
[2] [SY14]

Simulating linear-time 2nfa(k)'s in NTISP($n^2/\log(n), \log n$)

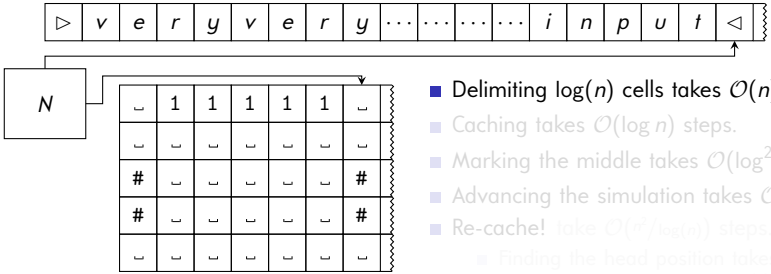


- Delimiting $\log(n)$ cells takes $\mathcal{O}(n)$ steps.
- Caching takes $\mathcal{O}(\log n)$ steps.
- Marking the middle takes $\mathcal{O}(\log^2 n)$ steps.
- Advancing the simulation takes $\mathcal{O}(n)$ steps.
- Re-cache! take $\mathcal{O}(n^2/\log(n))$ steps.
  - Finding the head position takes $\mathcal{O}(n)$ steps.
  - No re-cache for the next $\log(n)/2$ steps.
  - Linear runtime, thus $\mathcal{O}(n/\log(n))$ re-caches.

Thanks to Martin Kutrib.

M. Utkan Gezer [utkan.gezer@boun.edu.tr], A.C. Cem Say
Department of Computer Engineering, Boğaziçi University, Bebek, 34342 Istanbul, Turkey

Simulating linear-time 2nfa(k)'s in NTISP($n^2/\log(n), \log n$)

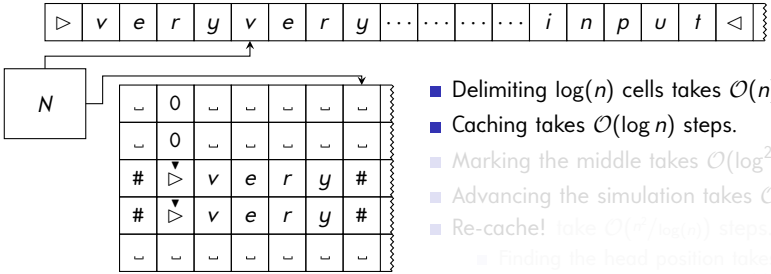


- Delimiting $\log(n)$ cells takes $\mathcal{O}(n)$ steps.
- Caching takes $\mathcal{O}(\log n)$ steps.
- Marking the middle takes $\mathcal{O}(\log^2 n)$ steps.
- Advancing the simulation takes $\mathcal{O}(n)$ steps.
- Re-cache! take $\mathcal{O}(n^2/\log(n))$ steps.
  - Finding the head position takes $\mathcal{O}(n)$ steps.
  - No re-cache for the next $\log(n)/2$ steps.
  - Linear runtime, thus $\mathcal{O}(n/\log(n))$ re-caches.

Thanks to Martin Kutrib.

M. Utkan Gezer [utkan.gezer@boun.edu.tr], A.C. Cem Say
Department of Computer Engineering, Boğaziçi University, Bebek, 34342 Istanbul, Turkey

Simulating linear-time 2nfa(k)'s in NTISP($n^2/\log(n), \log n$)

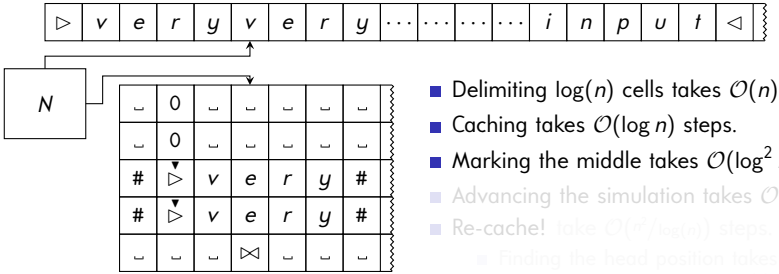


- Delimiting $\log(n)$ cells takes $\mathcal{O}(n)$ steps.
- Caching takes $\mathcal{O}(\log n)$ steps.
- Marking the middle takes $\mathcal{O}(\log^2 n)$ steps.
- Advancing the simulation takes $\mathcal{O}(n)$ steps.
- Re-cache! take $\mathcal{O}(n^2/\log(n))$ steps.
  - Finding the head position takes $\mathcal{O}(n)$ steps.
  - No re-cache for the next $\log(n)/2$ steps.
  - Linear runtime, thus $\mathcal{O}(n/\log(n))$ re-caches.

Thanks to Martin Kutrib.

M. Utkan Gezer [utkan.gezer@boun.edu.tr], A.C. Cem Say
Department of Computer Engineering, Boğaziçi University, Bebek, 34342 Istanbul, Turkey

Simulating linear-time 2nfa(k)'s in NTISP($n^2/\log(n), \log n$)

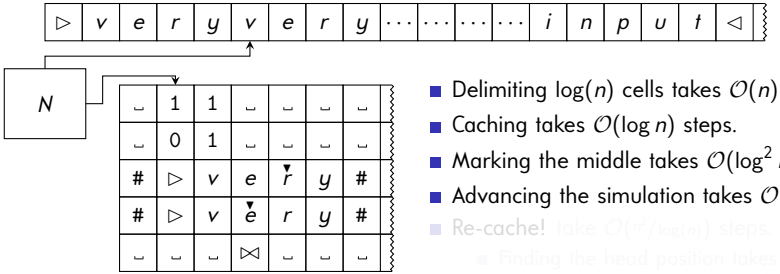


- Delimiting $\log(n)$ cells takes $\mathcal{O}(n)$ steps.
- Caching takes $\mathcal{O}(\log n)$ steps.
- Marking the middle takes $\mathcal{O}(\log^2 n)$ steps.
- Advancing the simulation takes $\mathcal{O}(n)$ steps.
- Re-cache! take $\mathcal{O}(n^2/\log(n))$ steps.
  - Finding the head position takes $\mathcal{O}(n)$ steps.
  - No re-cache for the next $\log(n)/2$ steps.
  - Linear runtime, thus $\mathcal{O}(n/\log(n))$ re-caches.

Thanks to Martin Kutrib.

M. Utkan Gezer [utkan.gezer@boun.edu.tr], A.C. Cem Say
Department of Computer Engineering, Boğaziçi University, Bebek, 34342 Istanbul, Turkey

Simulating linear-time 2nfa(k)'s in NTISP($n^2/\log(n), \log n$)

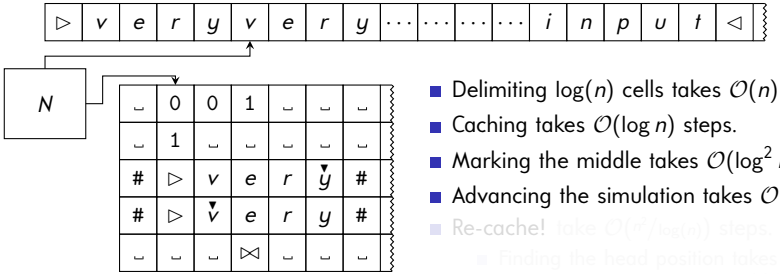


- Delimiting $\log(n)$ cells takes $\mathcal{O}(n)$ steps.
- Caching takes $\mathcal{O}(\log n)$ steps.
- Marking the middle takes $\mathcal{O}(\log^2 n)$ steps.
- Advancing the simulation takes $\mathcal{O}(n)$ steps.
- Re-cache! take $\mathcal{O}(n^2/\log(n))$ steps.
  - Finding the head position takes $\mathcal{O}(n)$ steps.
  - No re-cache for the next $\log(n)/2$ steps.
  - Linear runtime, thus $\mathcal{O}(n/\log(n))$ re-caches.

Thanks to Martin Kutrib.

M. Utkan Gezer [utkan.gezer@boun.edu.tr], A.C. Cem Say
Department of Computer Engineering, Boğaziçi University, Bebek, 34342 Istanbul, Turkey

Simulating linear-time 2nfa(k)'s in NTISP($n^2/\log(n), \log n$)

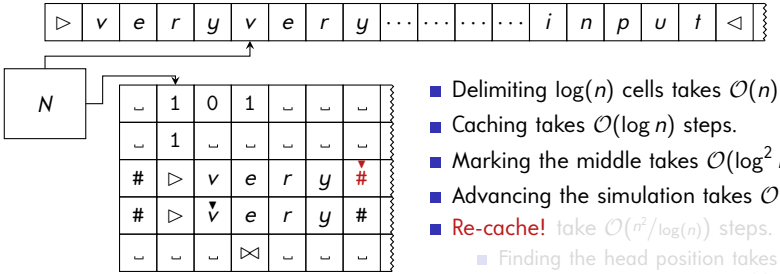


- Delimiting $\log(n)$ cells takes $\mathcal{O}(n)$ steps.
- Caching takes $\mathcal{O}(\log n)$ steps.
- Marking the middle takes $\mathcal{O}(\log^2 n)$ steps.
- Advancing the simulation takes $\mathcal{O}(n)$ steps.
- Re-cache! take $\mathcal{O}(n^2/\log(n))$ steps.
  - Finding the head position takes $\mathcal{O}(n)$ steps.
  - No re-cache for the next $\log(n)/2$ steps.
  - Linear runtime, thus $\mathcal{O}(n/\log(n))$ re-caches.

Thanks to Martin Kutrib.

M. Utkan Gezer [utkan.gezer@boun.edu.tr], A.C. Cem Say
Department of Computer Engineering, Boğaziçi University, Bebek, 34342 Istanbul, Turkey

Simulating linear-time 2nfa(k)'s in NTISP($n^2/\log(n), \log n$)

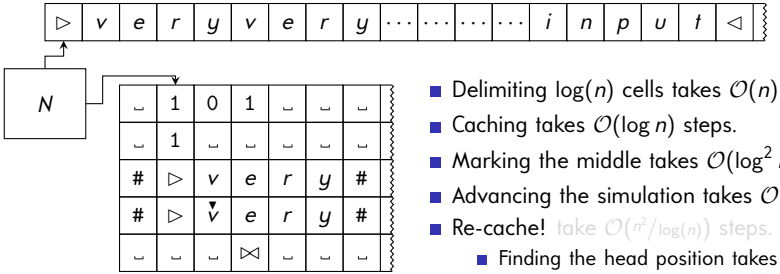


- Delimiting $\log(n)$ cells takes $\mathcal{O}(n)$ steps.
- Caching takes $\mathcal{O}(\log n)$ steps.
- Marking the middle takes $\mathcal{O}(\log^2 n)$ steps.
- Advancing the simulation takes $\mathcal{O}(n)$ steps.
- Re-cache! take $\mathcal{O}(n^2/\log(n))$ steps.
  - Finding the head position takes $\mathcal{O}(n)$ steps.
  - No re-cache for the next $\log(n)/2$ steps.
  - Linear runtime, thus $\mathcal{O}(n/\log(n))$ re-caches.

Thanks to Martin Kutrib.

M. Utkan Gezer [utkan.gezer@boun.edu.tr], A.C. Cem Say

Simulating linear-time 2nfa(k)'s in NTISP($n^2/\log(n)$, $\log n$)

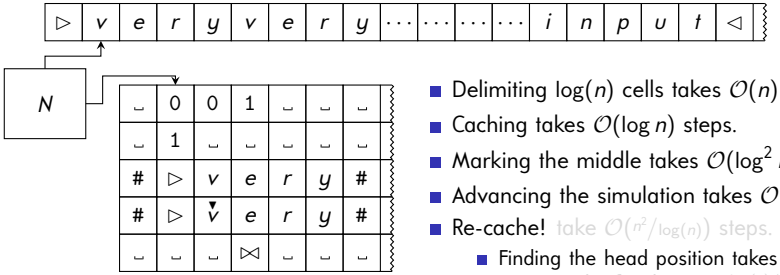


- Delimiting $\log(n)$ cells takes $\mathcal{O}(n)$ steps.
- Caching takes $\mathcal{O}(\log n)$ steps.
- Marking the middle takes $\mathcal{O}(\log^2 n)$ steps.
- Advancing the simulation takes $\mathcal{O}(n)$ steps.
- Re-cache! take $\mathcal{O}(n^2/\log(n))$ steps.
  - Finding the head position takes $\mathcal{O}(n)$ steps.
  - No re-cache for the next $\log(n)/2$ steps.
  - Linear runtime, thus $\mathcal{O}(n/\log(n))$ re-caches.

Thanks to Martin Kutrib.

M. Utkan Gezer [utkan.gezer@boun.edu.tr], A.C. Cem Say

Simulating linear-time 2nfa(k)'s in NTISP($n^2/\log(n), \log n$)

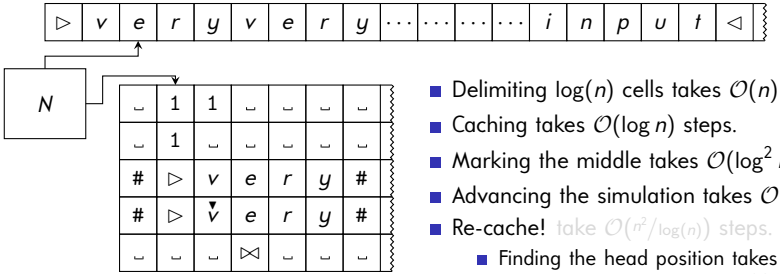


- Delimiting $\log(n)$ cells takes $\mathcal{O}(n)$ steps.
- Caching takes $\mathcal{O}(\log n)$ steps.
- Marking the middle takes $\mathcal{O}(\log^2 n)$ steps.
- Advancing the simulation takes $\mathcal{O}(n)$ steps.
- Re-cache! take $\mathcal{O}(n^2/\log(n))$ steps.
  - Finding the head position takes $\mathcal{O}(n)$ steps.
  - No re-cache for the next $\log(n)/2$ steps.
  - Linear runtime, thus $\mathcal{O}(n/\log(n))$ re-caches.

Thanks to Martin Kutrib.

M. Utkan Gezer [utkan.gezer@boun.edu.tr], A.C. Cem Say

Simulating linear-time 2nfa(k)'s in NTISP($n^2/\log(n), \log n$)

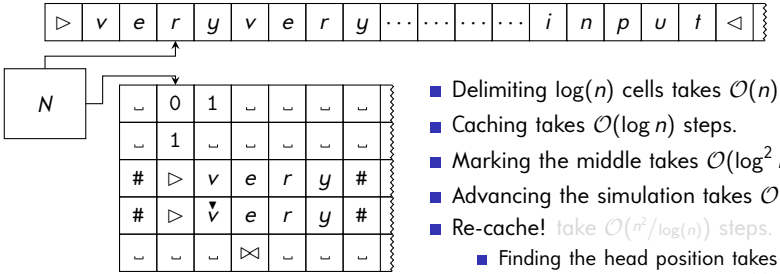


- Delimiting $\log(n)$ cells takes $\mathcal{O}(n)$ steps.
- Caching takes $\mathcal{O}(\log n)$ steps.
- Marking the middle takes $\mathcal{O}(\log^2 n)$ steps.
- Advancing the simulation takes $\mathcal{O}(n)$ steps.
- Re-cache! take $\mathcal{O}(n^2/\log(n))$ steps.
  - Finding the head position takes $\mathcal{O}(n)$ steps.
  - No re-cache for the next $\log(n)/2$ steps.
  - Linear runtime, thus $\mathcal{O}(n/\log(n))$ re-caches.

Thanks to Martin Kutrib.

M. Utkan Gezer [utkan.gezer@boun.edu.tr], A.C. Cem Say

Simulating linear-time 2nfa(k)'s in NTISP($n^2/\log(n), \log n$)

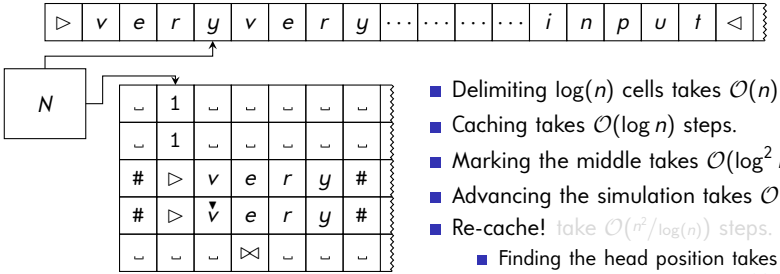


- Delimiting $\log(n)$ cells takes $\mathcal{O}(n)$ steps.
- Caching takes $\mathcal{O}(\log n)$ steps.
- Marking the middle takes $\mathcal{O}(\log^2 n)$ steps.
- Advancing the simulation takes $\mathcal{O}(n)$ steps.
- Re-cache! take $\mathcal{O}(n^2/\log(n))$ steps.
  - Finding the head position takes $\mathcal{O}(n)$ steps.
  - No re-cache for the next $\log(n)/2$ steps.
  - Linear runtime, thus $\mathcal{O}(n/\log(n))$ re-caches.

Thanks to Martin Kutrib.

M. Utkan Gezer [utkan.gezer@boun.edu.tr], A.C. Cem Say

Simulating linear-time 2nfa(k)'s in NTISP($n^2/\log(n), \log n$)

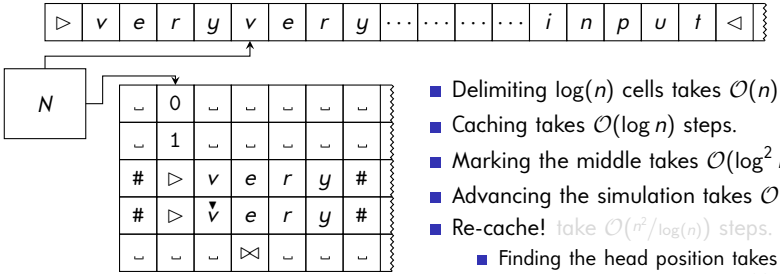


- Delimiting $\log(n)$ cells takes $\mathcal{O}(n)$ steps.
- Caching takes $\mathcal{O}(\log n)$ steps.
- Marking the middle takes $\mathcal{O}(\log^2 n)$ steps.
- Advancing the simulation takes $\mathcal{O}(n)$ steps.
- Re-cache! take $\mathcal{O}(n^2/\log(n))$ steps.
  - Finding the head position takes $\mathcal{O}(n)$ steps.
  - No re-cache for the next $\log(n)/2$ steps.
  - Linear runtime, thus $\mathcal{O}(n/\log(n))$ re-caches.

Thanks to Martin Kutrib.

M. Utkan Gezer [utkan.gezer@boun.edu.tr], A.C. Cem Say

Simulating linear-time 2nfa(k)'s in NTISP($n^2/\log(n), \log n$)

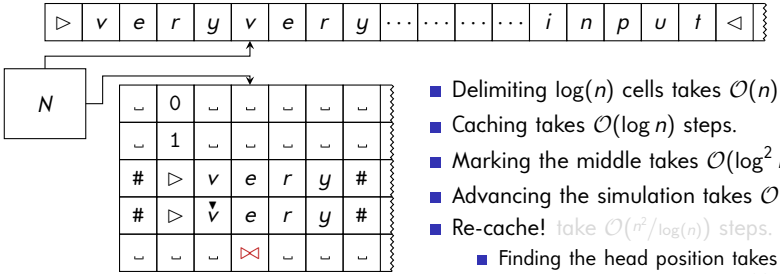


- Delimiting $\log(n)$ cells takes $\mathcal{O}(n)$ steps.
- Caching takes $\mathcal{O}(\log n)$ steps.
- Marking the middle takes $\mathcal{O}(\log^2 n)$ steps.
- Advancing the simulation takes $\mathcal{O}(n)$ steps.
- Re-cache! take $\mathcal{O}(n^2/\log(n))$ steps.
  - Finding the head position takes $\mathcal{O}(n)$ steps.
  - No re-cache for the next $\log(n)/2$ steps.
  - Linear runtime, thus $\mathcal{O}(n/\log(n))$ re-caches.

Thanks to Martin Kutrib.

M. Utkan Gezer [utkan.gezer@boun.edu.tr], A.C. Cem Say

Simulating linear-time 2nfa(k)'s in NTISP($n^2/\log(n), \log n$)

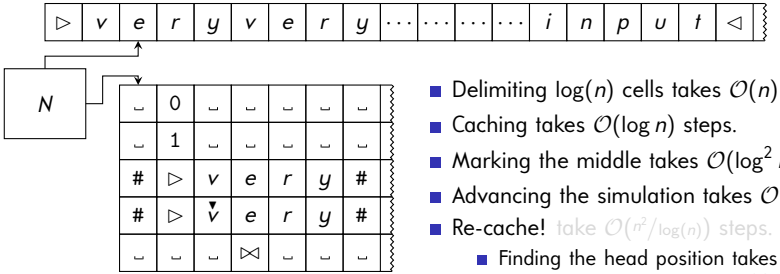


- Delimiting $\log(n)$ cells takes $\mathcal{O}(n)$ steps.
- Caching takes $\mathcal{O}(\log n)$ steps.
- Marking the middle takes $\mathcal{O}(\log^2 n)$ steps.
- Advancing the simulation takes $\mathcal{O}(n)$ steps.
- Re-cache! take $\mathcal{O}(n^2/\log(n))$ steps.
  - Finding the head position takes $\mathcal{O}(n)$ steps.
  - No re-cache for the next $\log(n)/2$ steps.
  - Linear runtime, thus $\mathcal{O}(n/\log(n))$ re-caches.

Thanks to Martin Kutrib.

M. Utkan Gezer [utkan.gezer@boun.edu.tr], A.C. Cem Say

Simulating linear-time 2nfa(k)'s in NTISP($n^2/\log(n), \log n$)

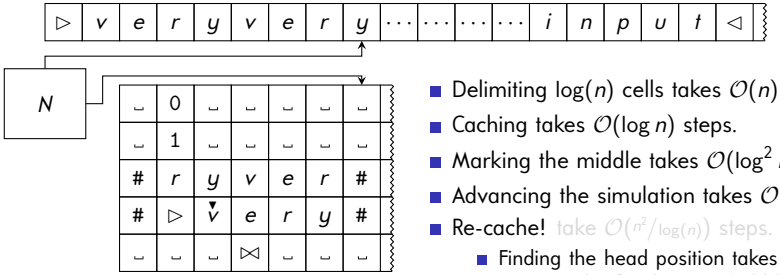


- Delimiting $\log(n)$ cells takes $\mathcal{O}(n)$ steps.
- Caching takes $\mathcal{O}(\log n)$ steps.
- Marking the middle takes $\mathcal{O}(\log^2 n)$ steps.
- Advancing the simulation takes $\mathcal{O}(n)$ steps.
- Re-cache! take $\mathcal{O}(n^2/\log(n))$ steps.
  - Finding the head position takes $\mathcal{O}(n)$ steps.
  - No re-cache for the next $\log(n)/2$ steps.
  - Linear runtime, thus $\mathcal{O}(n/\log(n))$ re-caches.

Thanks to Martin Kutrib.

M. Utkan Gezer [utkan.gezer@boun.edu.tr], A.C. Cem Say

Simulating linear-time 2nfa(k)'s in NTISP($n^2/\log(n)$, $\log n$)

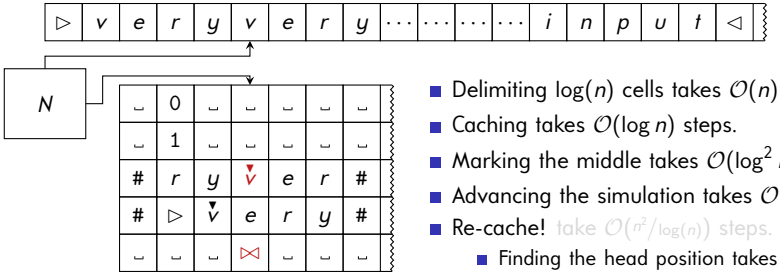


- Delimiting $\log(n)$ cells takes $\mathcal{O}(n)$ steps.
- Caching takes $\mathcal{O}(\log n)$ steps.
- Marking the middle takes $\mathcal{O}(\log^2 n)$ steps.
- Advancing the simulation takes $\mathcal{O}(n)$ steps.
- Re-cache! take $\mathcal{O}(n^2/\log(n))$ steps.
  - Finding the head position takes $\mathcal{O}(n)$ steps.
  - No re-cache for the next $\log(n)/2$ steps.
  - Linear runtime, thus $\mathcal{O}(n/\log(n))$ re-caches.

Thanks to Martin Kutrib.

M. Utkan Gezer [utkan.gezer@boun.edu.tr], A.C. Cem Say

Simulating linear-time 2nfa(k)'s in NTISP($n^2/\log(n), \log n$)

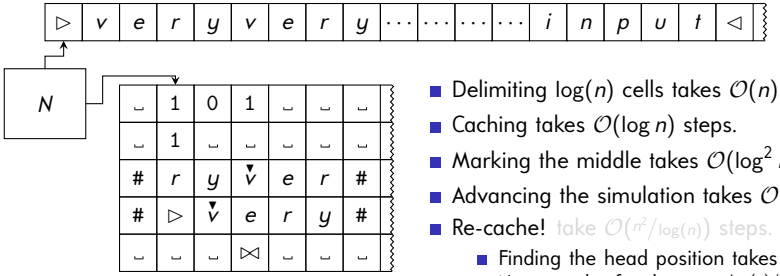


- Delimiting $\log(n)$ cells takes $\mathcal{O}(n)$ steps.
- Caching takes $\mathcal{O}(\log n)$ steps.
- Marking the middle takes $\mathcal{O}(\log^2 n)$ steps.
- Advancing the simulation takes $\mathcal{O}(n)$ steps.
- Re-cache! take $\mathcal{O}(n^2/\log(n))$ steps.
  - Finding the head position takes $\mathcal{O}(n)$ steps.
  - No re-cache for the next $\log(n)/2$ steps.
  - Linear runtime, thus $\mathcal{O}(n/\log(n))$ re-caches.

Thanks to Martin Kutrib.

M. Utkan Gezer [utkan.gezer@boun.edu.tr], A.C. Cem Say

Simulating linear-time 2nfa(k)'s in NTISP($n^2/\log(n)$, $\log n$)

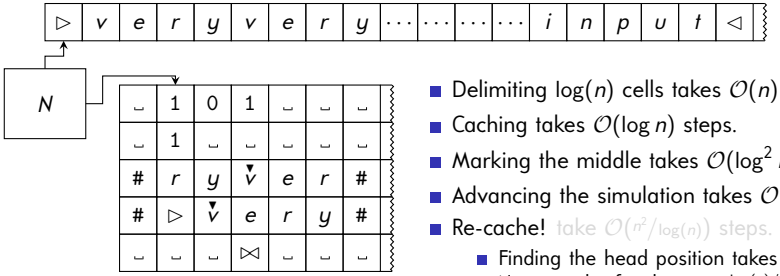


- Delimiting $\log(n)$ cells takes $\mathcal{O}(n)$ steps.
- Caching takes $\mathcal{O}(\log n)$ steps.
- Marking the middle takes $\mathcal{O}(\log^2 n)$ steps.
- Advancing the simulation takes $\mathcal{O}(n)$ steps.
- Re-cache! take $\mathcal{O}(n^2/\log(n))$ steps.
  - Finding the head position takes $\mathcal{O}(n)$ steps.
  - No re-cache for the next $\log(n)/2$ steps.
  - Linear runtime, thus $\mathcal{O}(n/\log(n))$ re-caches.

Thanks to Martin Kutrib.

M. Utkan Gezer [utkan.gezer@boun.edu.tr], A.C. Cem Say

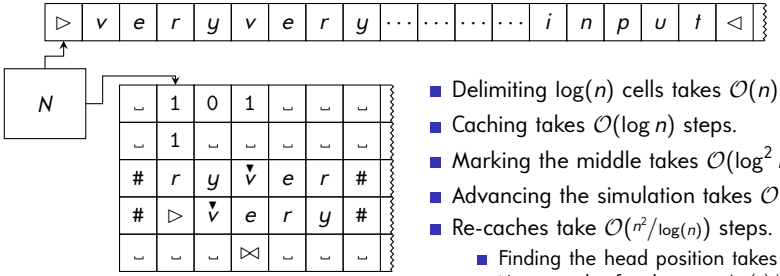
Simulating linear-time 2nfa(k)'s in NTISP($n^2/\log(n)$, $\log n$)



- Delimiting $\log(n)$ cells takes $\mathcal{O}(n)$ steps.
- Caching takes $\mathcal{O}(\log n)$ steps.
- Marking the middle takes $\mathcal{O}(\log^2 n)$ steps.
- Advancing the simulation takes $\mathcal{O}(n)$ steps.
- Re-cache! take $\mathcal{O}(n^2/\log(n))$ steps.
  - Finding the head position takes $\mathcal{O}(n)$ steps.
  - No re-cache for the next $\log(n)/2$ steps.
  - Linear runtime, thus $\mathcal{O}(n/\log(n))$ re-caches.

Thanks to Martin Kutrib.

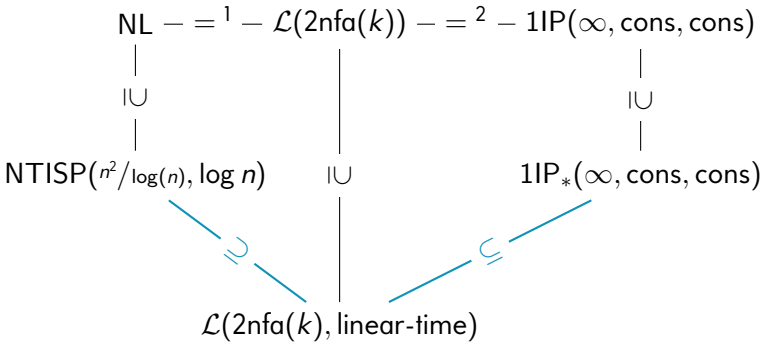
Simulating linear-time 2nfa(k)'s in NTISP($n^2/\log(n), \log n$)



- Delimiting $\log(n)$ cells takes $\mathcal{O}(n)$ steps.
- Caching takes $\mathcal{O}(\log n)$ steps.
- Marking the middle takes $\mathcal{O}(\log^2 n)$ steps.
- Advancing the simulation takes $\mathcal{O}(n)$ steps.
- Re-caches take $\mathcal{O}(n^2/\log(n))$ steps.
  - Finding the head position takes $\mathcal{O}(n)$ steps.
  - No re-cache for the next $\log(n)/2$ steps.
  - Linear runtime, thus $\mathcal{O}(n/\log(n))$ re-caches.

Thanks to Martin Kutrib.

Results and open questions



[1] [Har72]
[2] [SY14]

M. Utkan Gezer [utkan.gezer@boun.edu.tr], A.C. Cem Say

References

[GS21] M. Utkan Gezer and A.C. Cem Say. “Constant-space, constant-randomness verifiers with arbitrarily small error”. In: *Information and Computation* (2021), p. 104744.

[Har72] Juris Hartmanis. “On non-determinancy in simple computing devices”. In: *Acta Informatica* 1.4 (1972), pp. 336–344.

[Mon80] Burkhard Monien. “Two-way multihead automata over a one-letter alphabet”. In: *RAIRO. Inform. théor.* 14.1 (1980), pp. 67–82.

[SY14] A. C. Cem Say and Abuzer Yakaryılmaz. “Finite state verifiers with constant randomness”. In: *Logical Methods in Computer Science* 10.3 (Aug. 2014).