Real-Time, Constant-Space, Constant-Randomness Verifiers

Özdeniz Dolu Nevzat Ersoy M. Utkan Gezer A.C. Cem Say

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

CIAA 2022

Table of Contents

Multi-head finite automata

$1\mathrm{nfa}(k)$ nondeterministic finite automaton with k read-only heads that cannot go left

$1\mathrm{dfa}(k)$ deterministic variant

$\mathtt{1NFA(k)}$ and $\mathtt{1DFA(k)}$ the respective language classes

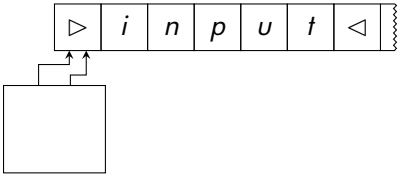
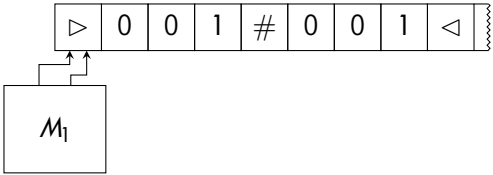


Figure: Depiction of a $1\mathrm{nfa}(2)$ (or $1\mathrm{dfa}(2)$)

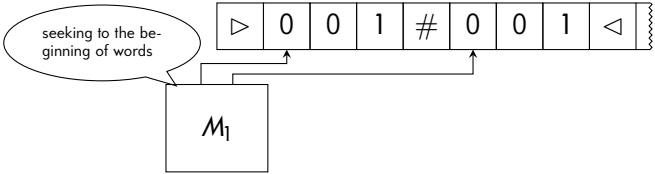
Multi-head finite automata examples

A $1\mathrm{dfa}(2)$ M_1 can recognize $L_{\mathrm{twin}} = \{\, w\#w \mid w \in \{\,0,1\,\}^* \,\}$.
Example execution of M_1 on 001#001:



Multi-head finite automata examples

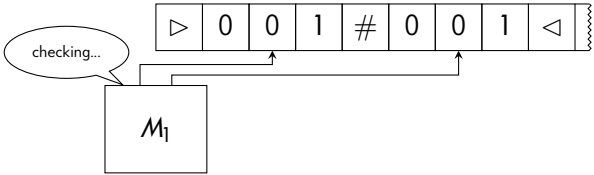
A $1\mathrm{dfa}(2)$ M_1 can recognize $L_{\mathrm{twin}} = \{\, w\#w \mid w \in \{\,0,1\,\}^* \,\}$.
Example execution of M_1 on 001#001:



Ö. Dolu, N. Ersoy, M.U. Gezer [utkan.gezer@boun.edu.tr], A.C.C. Say

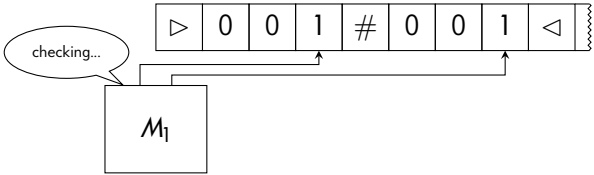
Multi-head finite automata examples

A $1\mathrm{dfa}(2)$ M_1 can recognize $L_{\mathrm{twin}} = \{\, w\#w \mid w \in \{\, 0,1 \,\}^* \,\}$.
Example execution of M_1 on 001#001:



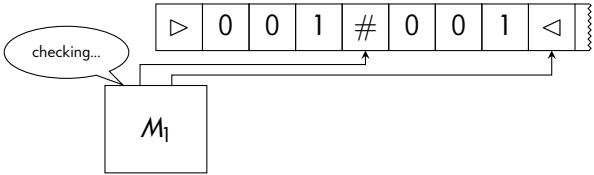
Multi-head finite automata examples

A $1\mathrm{dfa}(2)$ M_1 can recognize $L_{\mathrm{twin}} = \{\, w\#w \mid w \in \{\,0,1\,\}^* \,\}$.
Example execution of M_1 on 001#001:



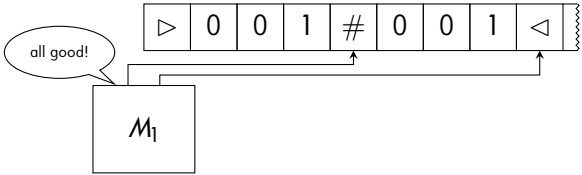
Multi-head finite automata examples

A $1\mathrm{dfa}(2)$ M_1 can recognize $L_{\mathrm{twin}} = \{\, w\#w \mid w \in \{\,0,1\,\}^* \,\}$.
Example execution of M_1 on 001#001:



Multi-head finite automata examples

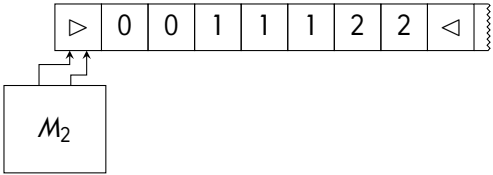
A $1\mathrm{dfa}(2)$ M_1 can recognize $L_{\mathrm{twin}} = \{\, w\#w \mid w \in \{\,0,1\,\}^* \,\}$.
Example execution of M_1 on 001#001:



Ö. Dolu, N. Ersoy, M.U. Gezer [utkan.gezer@boun.edu.tr], A.C.C. Say

Multi-head finite automata examples

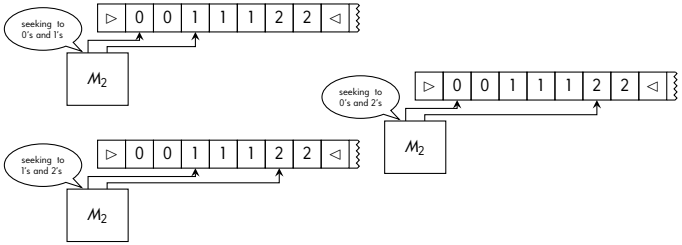
A $1\mathrm{nfa}(2)$ M_2 can recognize $L_{IK} = \left\{ 0^i 1^j 2^k \;\middle|\; i = j \text{ or } j = k \text{ or } i = k \right\}$.
Example execution of M_2 on $0^2 1^3 2^2$:



Ö. Dolu, N. Ersoy, M.U. Gezer [utkan.gezer@boun.edu.tr], A.C.C. Say

Multi-head finite automata examples

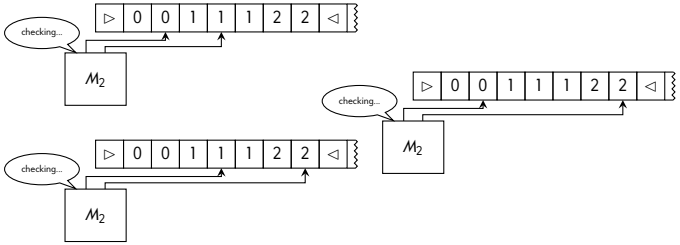
A $1\mathrm{nfa}(2)$ M_2 can recognize $L_{IK} = \left\{ 0^i 1^j 2^k \;\middle|\; i = j \text{ or } j = k \text{ or } i = k \right\}$.
Example execution of M_2 on $0^2 1^3 2^2$:



Ö. Dolu, N. Ersoy, M.U. Gezer [utkan.gezer@boun.edu.tr], A.C.C. Say

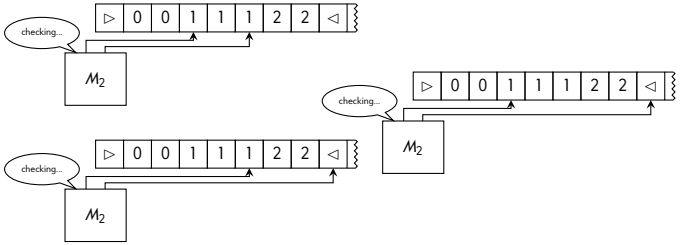
Multi-head finite automata examples

A $1\mathrm{nfa}(2)$ M_2 can recognize $L_{IK} = \left\{ 0^i 1^j 2^k \;\middle|\; i = j \text{ or } j = k \text{ or } i = k \right\}$.
Example execution of M_2 on $0^2 1^3 2^2$:



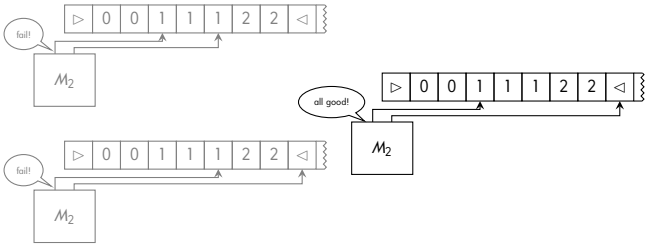
Multi-head finite automata examples

A $1\mathrm{nfa}(2)$ M_2 can recognize $L_{IK} = \left\{ 0^i 1^j 2^k \;\middle|\; i = j \text{ or } j = k \text{ or } i = k \right\}$.
Example execution of M_2 on $0^2 1^3 2^2$:



Multi-head finite automata examples

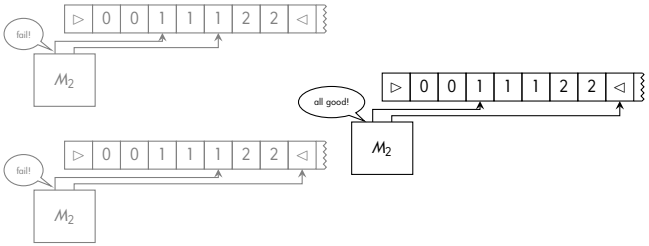
A $1\mathrm{nfa}(2)$ M_2 can recognize $L_{IK} = \left\{ 0^i 1^j 2^k \;\middle|\; i = j \text{ or } j = k \text{ or } i = k \right\}$.
Example execution of M_2 on $0^2 1^3 2^2$:



Ö. Dolu, N. Ersoy, M.U. Gezer [utkan.gezer@boun.edu.tr], A.C.C. Say

Multi-head finite automata examples

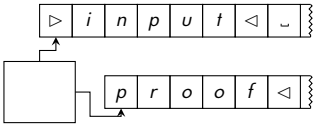
A $1\mathrm{nfa}(2)$ M_2 can recognize $L_{IK} = \{\, 0^i 1^j 2^k \mid i = j \text{ or } j = k \text{ or } i = k \,\}$.
Example execution of M_2 on $0^2 1^3 2^2$:
$L_{IK} \in \mathtt{1DFA}(3) \setminus \mathtt{1DFA}(2)$ [IK75].



Verifiers

A *verifier* is a finite automaton with one-way access to the most convincing (purported) proof of membership for its input (even when it is a nonmember).

$\mathtt{VER}(\mathtt{restriction}_1,\ldots,\mathtt{restriction}_k)$ class of languages verifiable under given restrictions

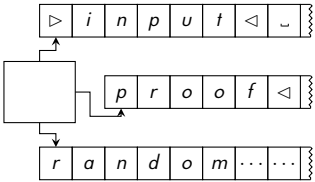


Ö. Dolu, N. Ersoy, M.U. Gezer [utkan.gezer@boun.edu.tr], A.C.C. Say

Verifiers

A *verifier* is a finite automaton with one-way access to the most convincing (purported) proof of membership for its input (even when it is a nonmember).

$\mathtt{VER}(\mathtt{restriction}_1,\ldots,\mathtt{restriction}_k)$ class of languages verifiable under given restrictions



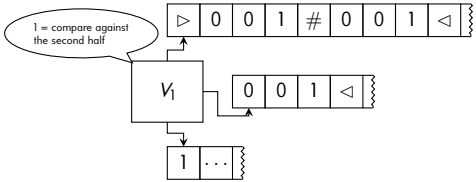
Ö. Dolu, N. Ersoy, M.U. Gezer [utkan.gezer@boun.edu.tr], A.C.C. Say

Error of a probabilistic verifier

- We aim for perfect completeness (= accepting members with probability 1).
- The error of a verifier is its maximum rate of failure to reject any nonmember input.

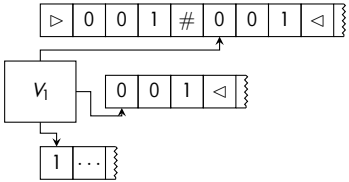
Verifier example

A verifier V_1 can recognize L_{twin} in real-time using constant-space and a single random bit by asking for a proof that provides one of the (purportedly repeated) words.



Verifier example

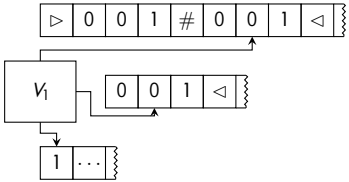
A verifier V_1 can recognize L_{twin} in real-time using constant-space and a single random bit by asking for a proof that provides one of the (purportedly repeated) words.



Ö. Dolu, N. Ersoy, M.U. Gezer [utkan.gezer@boun.edu.tr], A.C.C. Say

Verifier example

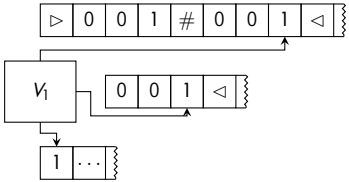
A verifier V_1 can recognize L_{twin} in real-time using constant-space and a single random bit by asking for a proof that provides one of the (purportedly repeated) words.



Ö. Dolu, N. Ersoy, M.U. Gezer [utkan.gezer@boun.edu.tr], A.C.C. Say

Verifier example

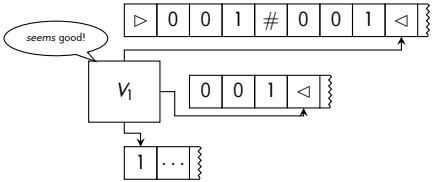
A verifier V_1 can recognize L_{twin} in real-time using constant-space and a single random bit by asking for a proof that provides one of the (purportedly repeated) words.



Ö. Dolu, N. Ersoy, M.U. Gezer [utkan.gezer@boun.edu.tr], A.C.C. Say

Verifier example

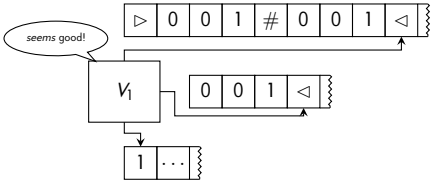
A verifier V_1 can recognize L_{twin} in real-time using constant-space and a single random bit by asking for a proof that provides one of the (purportedly repeated) words.



Ö. Dolu, N. Ersoy, M.U. Gezer [utkan.gezer@boun.edu.tr], A.C.C. Say

Verifier example

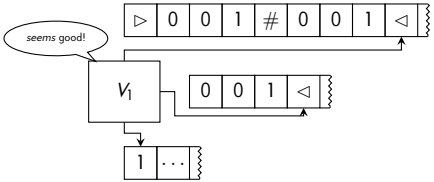
A verifier V_1 can recognize L_{twin} in real-time using constant-space and a single random bit by asking for a proof that provides one of the (purportedly repeated) words.
What if the first half was not in match?
Random bits are hidden from the prover, so there is a 50% chance of being caught.



Ö. Dolu, N. Ersoy, M.U. Gezer [utkan.gezer@boun.edu.tr], A.C.C. Say

Verifier example

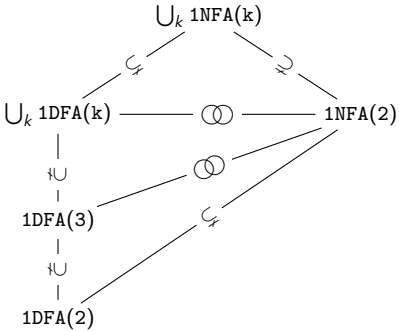
A verifier V_1 can recognize L_{twin} in real-time using constant-space and a single random bit by asking for a proof that provides one of the (purportedly repeated) words.
What if the first half was not in match?
Random bits are hidden from the prover, so there is a 50% chance of being caught.



Ö. Dolu, N. Ersoy, M.U. Gezer [utkan.gezer@boun.edu.tr], A.C.C. Say

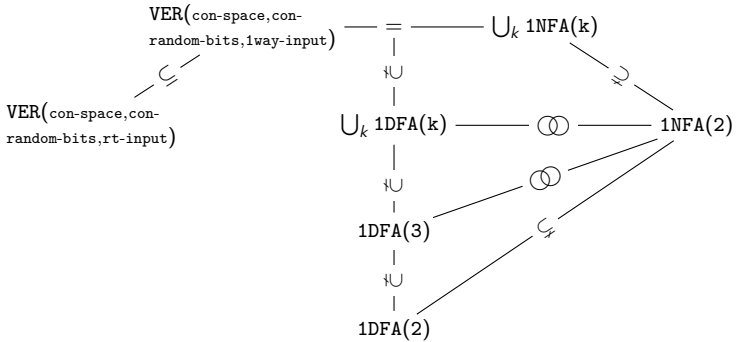
Table of Contents

What was known, our motivation, and our results



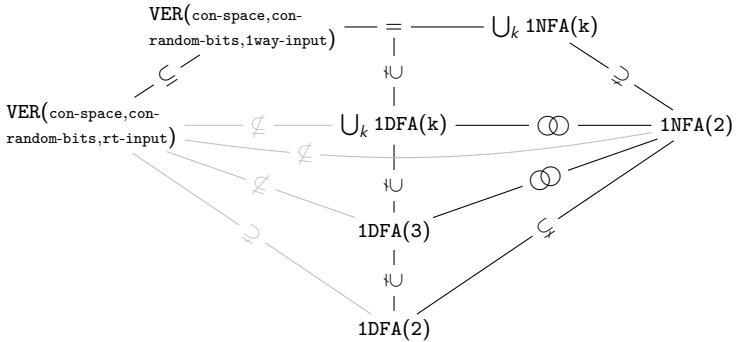
[HKM11]

What was known, our motivation, and our results



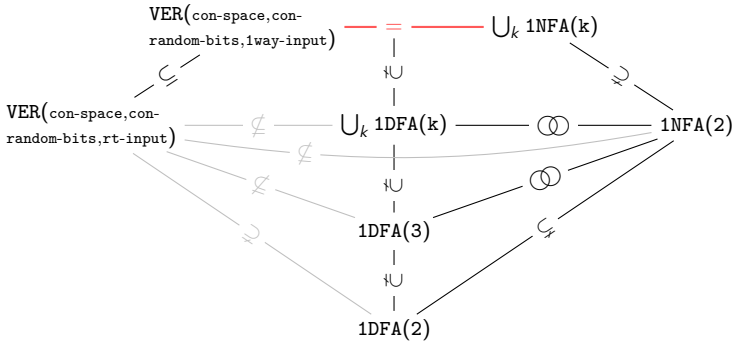
[HKM11]
[SY14]

What was known, our motivation, and our results



[HKM11]
[SY14]

What was known, our motivation, and our results



[HKM11]
[SY14]

Table of Contents

One-way verification equals $\bigcup_k$ 1NFA(k)

Theorem 1

$\mathtt{VER}(\mathtt{con\text{-}space},\mathtt{con\text{-}random\text{-}bits},\mathtt{1way\text{-}input}) = \bigcup_{\mathtt{k}} \mathtt{1NFA}(\mathtt{k})$.

Verifying what is recognized

- Verifier expects the trace of an accepting computation of the $1\mathrm{nfa}(k)$ as a proof:
  - what is read by each head
  - which nondeterministic step is taken
- Verifier randomly (using $\lceil \log(k+1) \rceil$ coins) does either one of the following:
  - chooses a head at random and verifies its readings along the trace
  - keeps a timer for the runtime limit of $1\mathrm{nfa}(k)$ (about $n \cdot k$)

Ö. Dolu, N. Ersoy, M.U. Gezer [utkan.gezer@boun.edu.tr], A.C.C. Say

One-way verification equals $\bigcup_k$ 1NFA(k)

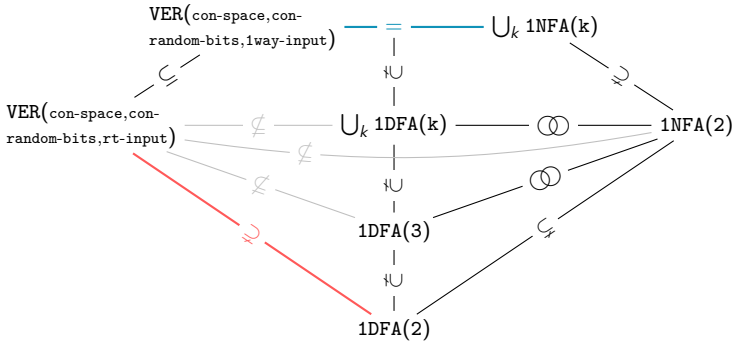
Theorem 1

$\mathtt{VER}(\mathtt{con\text{-}space},\mathtt{con\text{-}random\text{-}bits},\mathtt{1way\text{-}input}) = \bigcup_{\mathtt{k}} \mathtt{1NFA(k)}$.

Recognizing what is verified

- Verifier can be modified to make all its r coin flips at the beginning and then transfer execution to corresponding *deterministic verifier* (DV)
- A $1\mathrm{nfa}(2^r)$ M can simulate all the DVs at once:
  - each head follows a DV's head
  - M guesses a proof symbol and simulates each DV with it until they halt or consume the symbol
  - this process is repeated until a DV rejects or all the DVs accept

Overview



[HKM11]
[SY14]

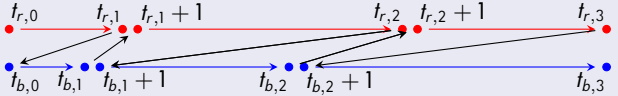
Real-time verification of 1DFA(2)

Theorem 2

$\texttt{1DFA(2)} \subseteq \texttt{VER}(\texttt{con-space},\texttt{con-random-bits},\texttt{rt-input})$.

Proof idea.

- The 1dfa(2) M can be modified to move exactly one of its heads at a time



- Verifier V asks for a proof that states the following four bits of information at each symbol:
  - state M was in and symbol r was reading while transitioning out of $t_{r,i}$
  - state M was in and symbol b was reading while transitioning out of $t_{b,i}$

Ö. Dolu, N. Ersoy, M.U. Gezer [utkan.gezer@boun.edu.tr], A.C.C. Say Department of Computer Engineering, Boğaziçi University, Bebek, 34342 Istanbul, Turkey

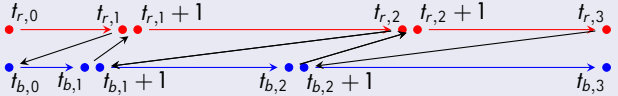
Real-time verification of 1DFA(2)

Theorem 2

$\texttt{1DFA(2)} \subseteq \texttt{VER}(\texttt{con-space}, \texttt{con-random-bits}, \texttt{rt-input})$.

Proof idea.

- The 1dfa(2) M can be modified to move exactly one of its heads at a time



- Intuitively, V;
  - "flash-sleeps" at $t_{x,i}$, verifying the proof as it does so
  - "wakes up" at $t_{x,i}+1$, reminded of the state of M by the proof

Ö. Dolu, N. Ersoy, M.U. Gezer [utkan.gezer@boun.edu.tr], A.C.C. Say

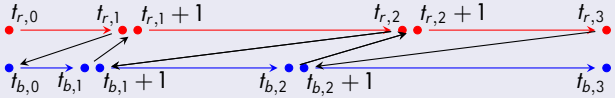
Real-time verification of 1DFA(2)

Theorem 2

$\texttt{1DFA(2)} \subseteq \texttt{VER}(\texttt{con-space},\texttt{con-random-bits},\texttt{rt-input})$.

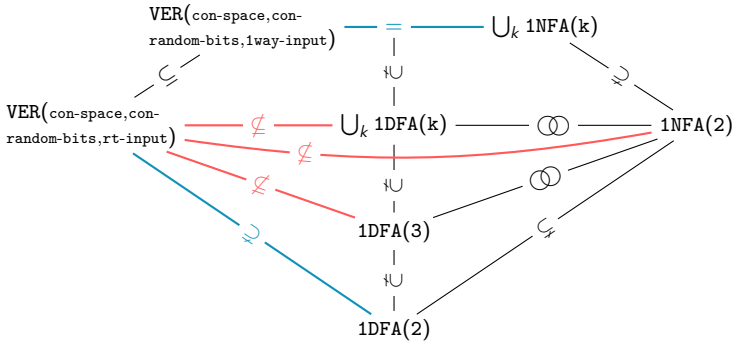
Proof idea.

- The $1\mathrm{dfa}(2)$ M can be modified to move exactly one of its heads at a time



- V chooses a head at random using a single random bit, validating the corresponding half of the proof as it flash-sleeps, while trusting the other half as it wakes up as it marches across the input in real-time

Overview



[HKM11]
[SY14]

Real-time verification of 1DFA(2)

Theorem 3

$\mathtt{VER}(\mathtt{con\text{-}space},\mathtt{con\text{-}random\text{-}bits},\mathtt{rt\text{-}input}) \setminus \bigcup_k \mathtt{1DFA}(k) \neq \emptyset$.

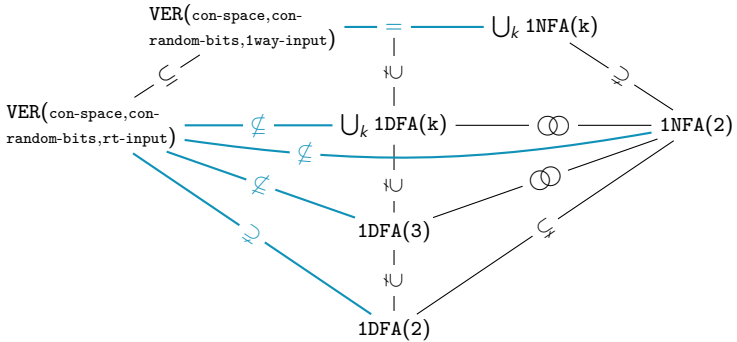
Proof idea.

- $L_{\text{nonpal}} = \{\, x\sigma y\sigma' z \mid x,y,z \in \{0,1\}^*; \sigma,\sigma' \in \{0,1\}; |x| = |z|; \sigma \neq \sigma' \,\}$ can be verified by a verifier V
- V asks for a proof that provides $|x|$ and $|y|$, e.g. $0^{|x|}10^{|y|}$
- On input w and proof 0^i10^j, V randomly either;
  - verifies that $w_{i+1} \neq w_{i+j+2}$ (thus $\sigma \neq \sigma'$)
  - verifies that $2i + j + 2 = |w|$ (thus $|x| = |z|$)

Ö. Dolu, N. Ersoy, M.U. Gezer [utkan.gezer@boun.edu.tr], A.C.C. Say Department of Computer Engineering, Boğaziçi University, Istanbul, Turkey

Table of Contents

Overview



[HKM11]
[SY14]

Conjectures

- $\mathtt{VER}(\mathtt{con\text{-}space},\mathtt{con\text{-}random\text{-}bits},\mathtt{rt\text{-}input}) \subsetneq \mathtt{VER}(\mathtt{con\text{-}space},\mathtt{con\text{-}random\text{-}bits},\mathtt{1way\text{-}input})$
- The following languages can be verified one-way, but seemingly not in real-time:

$$L_{\text{match}} = \left\{ x\#y_1\#y_2\#\cdots\#y_k \;\middle|\; \begin{array}{l} x, y_i \in \{0,1\}^+ \text{ for all } i, k > 0, \\ \text{and } y_i = x \text{ for some } i \end{array} \right\}$$

$$L_{\frac{1}{2}} = \{ ww \mid w \in \{0,1\}^* \}$$

$$L_{\frac{2}{3}} = \{ xwx \mid x, w \in \{0,1\}^* \text{ and } |x| = |w| \}$$

Ö. Dolu, N. Ersoy, M.U. Gezer [utkan.gezer@boun.edu.tr], A.C.C. Say

References

[HKM11] Markus Holzer, Martin Kutrib, and Andreas Malcher. “Complexity of multi-head finite automata: Origins and directions”. en. In: *Theoretical Computer Science* 412.1-2 (Jan. 2011), pp. 83–96. ISSN: 03043975. (Visited on 02/02/2019).

[IK75] Oscar H. Ibarra and Chul E. Kim. “On 3-head versus 2-head finite automata”. en. In: *Acta Informatica* 4.2 (1975), pp. 193–200. ISSN: 0001-5903, 1432-0525. (Visited on 03/01/2019).

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