Synchronizing Finite Automata

Lecture II: Algorithmic Issues

Mikhail Volkov

Ural Federal University / Hunter College

1. Recap

Deterministic finite automata (DFA): $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$.

- Q the state set
- Σ the input alphabet
- $\delta : Q \times \Sigma \to Q$ the transition function

$\mathscr{A}$ is called synchronizing if there exists a word $w \in \Sigma^*$ whose action resets $\mathscr{A}$, that is, leaves the automaton in one particular state no matter which state in Q it started at: $\delta(q, w) = \delta(q', w)$ for all $q, q' \in Q$.

$|Q\,.\,w| = 1$. Here $Q\,.\,v = \{\delta(q, v) \mid q \in Q\}$.

Any w with this property is a reset word for $\mathscr{A}$.

1. Recap

Deterministic finite automata (DFA): $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$.

- Q the state set
- Σ the input alphabet
- $\delta : Q \times \Sigma \to Q$ the transition function

$\mathscr{A}$ is called synchronizing if there exists a word $w \in \Sigma^*$ whose action resets $\mathscr{A}$, that is, leaves the automaton in one particular state no matter which state in Q it started at: $\delta(q, w) = \delta(q', w)$ for all $q, q' \in Q$.

$|Q \,.\, w| = 1$. Here $Q \,.\, v = \{\delta(q, v) \mid q \in Q\}$.

Any w with this property is a reset word for $\mathscr{A}$.

1. Recap

Deterministic finite automata (DFA): $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$.

- Q the state set
- Σ the input alphabet
- $\delta : Q \times \Sigma \to Q$ the transition function

$\mathscr{A}$ is called synchronizing if there exists a word $w \in \Sigma^*$ whose action resets $\mathscr{A}$, that is, leaves the automaton in one particular state no matter which state in Q it started at: $\delta(q, w) = \delta(q', w)$ for all $q, q' \in Q$.

$|Q\,.\,w| = 1$. Here $Q\,.\,v = \{\delta(q, v) \mid q \in Q\}$.

Any w with this property is a reset word for $\mathscr{A}$.

1. Recap

Deterministic finite automata (DFA): $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$.

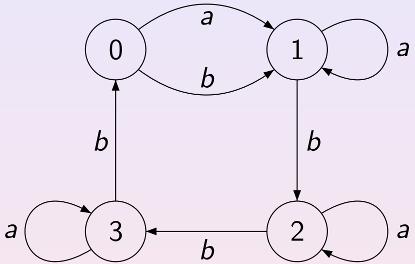
- Q the state set
- Σ the input alphabet
- $\delta : Q \times \Sigma \to Q$ the transition function

$\mathscr{A}$ is called synchronizing if there exists a word $w \in \Sigma^*$ whose action resets $\mathscr{A}$, that is, leaves the automaton in one particular state no matter which state in Q it started at: $\delta(q, w) = \delta(q', w)$ for all $q, q' \in Q$.

$|Q \,.\, w| = 1$. Here $Q \,.\, v = \{\delta(q, v) \mid q \in Q\}$.

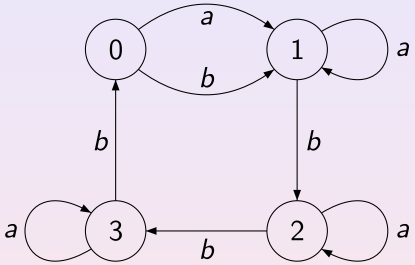
Any w with this property is a reset word for $\mathscr{A}$.

2. Example



A reset word is *abbbabbba*. In fact, we will see that this is the shortest reset word for this automaton.

2. Example



A reset word is *abbbabbba*. In fact, we will see that this is the shortest reset word for this automaton.

Not every DFA is synchronizing. Therefore, the very first question is the following one: *given an automaton, how to determine whether or not it is synchronizing?* This question is easy, and a straightforward solution comes from the classic powerset construction by Rabin and Scott.

The *power automaton* $\mathcal{P}(\mathscr{A})$ of a given DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$:
- states are the non-empty subsets of Q,
- $\delta(P, a) = P\,.\,a = \{\delta(p, a) \mid p \in P\}$

A $w \in \Sigma^*$ is a reset word for the DFA $\mathscr{A}$ iff w labels a path in $\mathcal{P}(\mathscr{A})$ starting at Q and ending at a singleton.

3. Power Automaton

Not every DFA is synchronizing. Therefore, the very first question is the following one: *given an automaton, how to determine whether or not it is synchronizing?* This question is easy, and a straightforward solution comes from the classic powerset construction by Rabin and Scott.

The *power automaton* $\mathcal{P}(\mathscr{A})$ of a given DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$:
- states are the non-empty subsets of Q,
- $\delta(P, a) = P\,.\,a = \{\delta(p, a) \mid p \in P\}$

A $w \in \Sigma^*$ is a reset word for the DFA $\mathscr{A}$ iff w labels a path in $\mathcal{P}(\mathscr{A})$ starting at Q and ending at a singleton.

Not every DFA is synchronizing. Therefore, the very first question is the following one: *given an automaton, how to determine whether or not it is synchronizing?* This question is easy, and a straightforward solution comes from the classic powerset construction by Rabin and Scott.

The *power automaton* $\mathcal{P}(\mathscr{A})$ of a given DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$:
- states are the non-empty subsets of Q,
- $\delta(P, a) = P \,.\, a = \{\delta(p, a) \mid p \in P\}$

A $w \in \Sigma^*$ is a reset word for the DFA $\mathscr{A}$ iff w labels a path in $\mathcal{P}(\mathscr{A})$ starting at Q and ending at a singleton.

Not every DFA is synchronizing. Therefore, the very first question is the following one: *given an automaton, how to determine whether or not it is synchronizing?* This question is easy, and a straightforward solution comes from the classic powerset construction by Rabin and Scott.

The *power automaton* $\mathcal{P}(\mathscr{A})$ of a given DFA $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$:
- states are the non-empty subsets of Q,
- $\delta(P, a) = P \,.\, a = \{\delta(p, a) \mid p \in P\}$

A $w \in \Sigma^*$ is a reset word for the DFA $\mathscr{A}$ iff w labels a path in $\mathcal{P}(\mathscr{A})$ starting at Q and ending at a singleton.

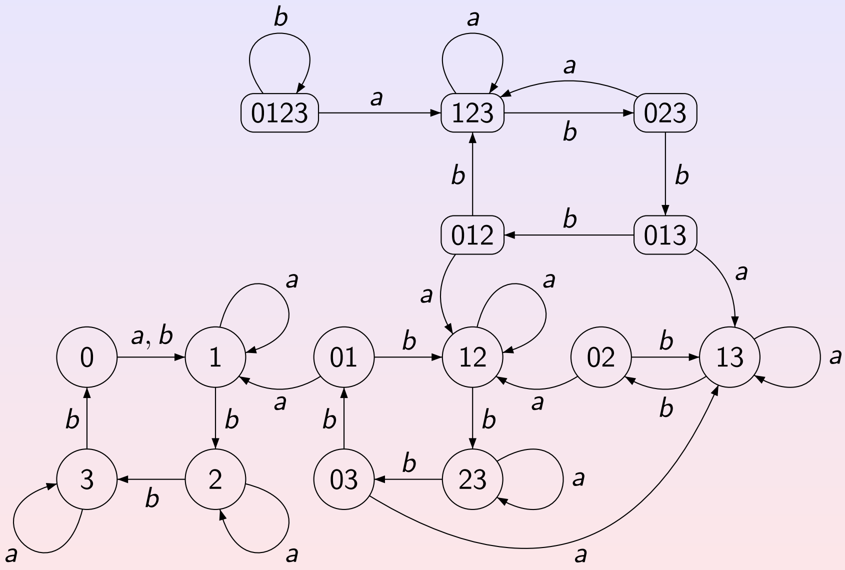
3. Power Automaton

Not every DFA is synchronizing. Therefore, the very first question is the following one: *given an automaton, how to determine whether or not it is synchronizing?* This question is easy, and a straightforward solution comes from the classic powerset construction by Rabin and Scott.

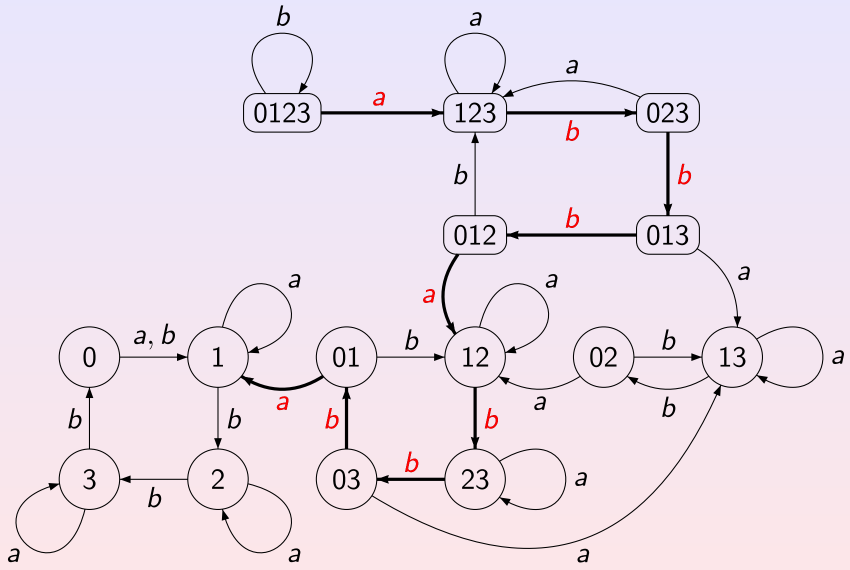
The *power automaton* $\mathcal{P}(\mathscr{A})$ of a given DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$:
- states are the non-empty subsets of Q,
- $\delta(P, a) = P\,.\,a = \{\delta(p, a) \mid p \in P\}$

A $w \in \Sigma^*$ is a reset word for the DFA $\mathscr{A}$ iff w labels a path in $\mathcal{P}(\mathscr{A})$ starting at Q and ending at a singleton.
Exercise: Write down a proof of this claim!

4. Example



4. Example



5. Polynomial Algorithm

Thus, the question of whether or not a given DFA $\mathscr{A}$ is synchronizing reduces to the following reachability question in the underlying digraph of the power automaton $\mathcal{P}(\mathscr{A})$: is there a path from Q to a singleton? The latter question can be easily answered by BFS. This algorithm is however exponential w.r.t. the size of $\mathscr{A}$.

The following result by Černý gives a polynomial algorithm:

Proposition. *A DFA* $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ *is synchronizing iff for every* $q, q' \in Q$ *there exists a word* $w \in \Sigma^*$ *such that* $\delta(q, w) = \delta(q', w)$.

Thus, the question of whether or not a given DFA $\mathscr{A}$ is synchronizing reduces to the following reachability question in the underlying digraph of the power automaton $\mathcal{P}(\mathscr{A})$: is there a path from Q to a singleton? The latter question can be easily answered by BFS. This algorithm is however exponential w.r.t. the size of $\mathscr{A}$.

The following result by Černý gives a polynomial algorithm:

Proposition. *A DFA* $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ *is synchronizing iff for every* $q, q' \in Q$ *there exists a word* $w \in \Sigma^*$ *such that* $\delta(q, w) = \delta(q', w)$.

5. Polynomial Algorithm

Thus, the question of whether or not a given DFA $\mathscr{A}$ is synchronizing reduces to the following reachability question in the underlying digraph of the power automaton $\mathcal{P}(\mathscr{A})$: is there a path from Q to a singleton? The latter question can be easily answered by BFS. This algorithm is however exponential w.r.t. the size of $\mathscr{A}$.

The following result by Černý gives a polynomial algorithm:

Proposition. *A DFA* $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ *is synchronizing iff for every* $q, q' \in Q$ *there exists a word* $w \in \Sigma^*$ *such that* $\delta(q, w) = \delta(q', w)$.

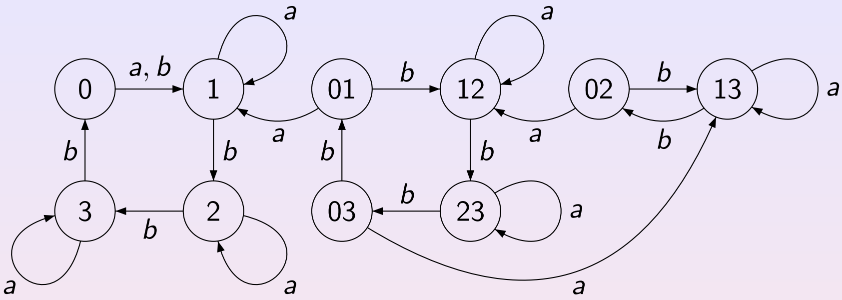
5. Polynomial Algorithm

Thus, the question of whether or not a given DFA $\mathscr{A}$ is synchronizing reduces to the following reachability question in the underlying digraph of the power automaton $\mathcal{P}(\mathscr{A})$: is there a path from Q to a singleton? The latter question can be easily answered by BFS. This algorithm is however exponential w.r.t. the size of $\mathscr{A}$.

The following result by Černý gives a polynomial algorithm:

Proposition. *A DFA* $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ *is synchronizing iff for every* $q, q' \in Q$ *there exists a word* $w \in \Sigma^*$ *such that* $\delta(q, w) = \delta(q', w)$.

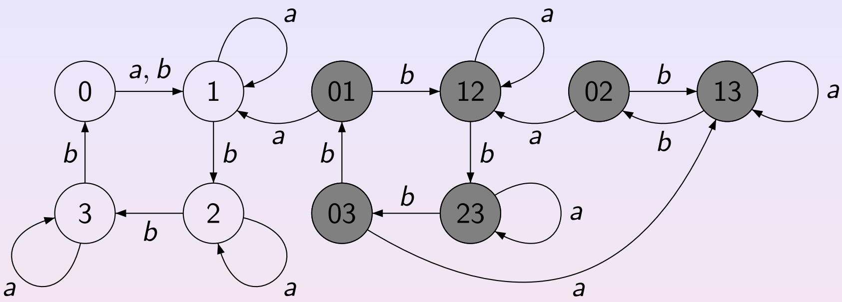
6. Example



a, $Q\,.\,a = \{1,2,3\}$; $a \cdot bba$, $Q\,.\,abba = \{1,3\}$

$abba \cdot babbba$, $Q\,.\,abbababbba = \{1\}$

Observe that the reset word constructed this way is of length 10 while we know a reset word of length 9.

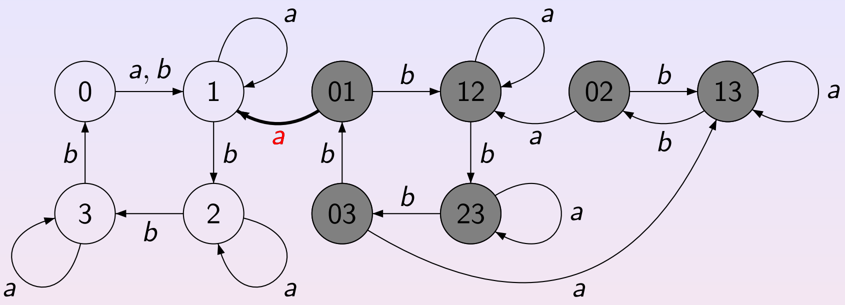


$a, \ Q\,.\,a = \{1,2,3\}; \quad a \cdot bba, \ Q\,.\,abba = \{1,3\}$

$abba \cdot babbba, \ Q\,.\,abbababbba = \{1\}$

Observe that the reset word constructed this way is of length 10 while we know a reset word of length 9.

6. Example

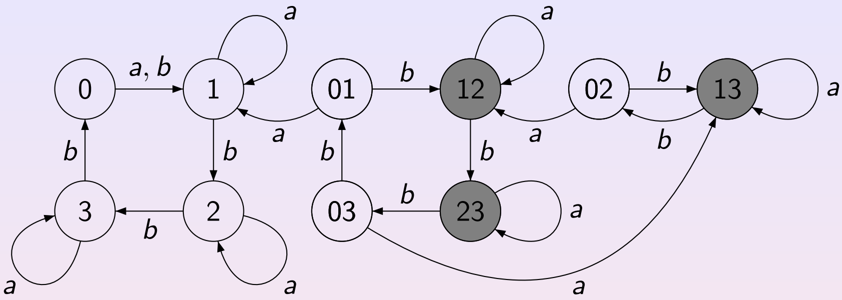


$a, \ Q\,.\,a = \{1, 2, 3\}; \quad a \cdot bba, \ Q\,.\,abba = \{1, 3\}$

$abba \cdot babbba, \ Q\,.\,abbababbba = \{1\}$

Observe that the reset word constructed this way is of length 10 while we know a reset word of length 9.

6. Example

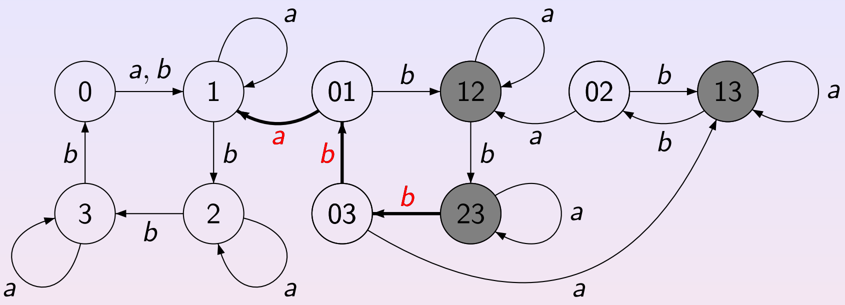


$a, \;\; Q\,.\,a = \{1,2,3\};$ $\quad a \cdot bba, \;\; Q\,.\,abba = \{1,3\}$

$abba \cdot babbba, \;\; Q\,.\,abbababbba = \{1\}$

Observe that the reset word constructed this way is of length 10 while we know a reset word of length 9.

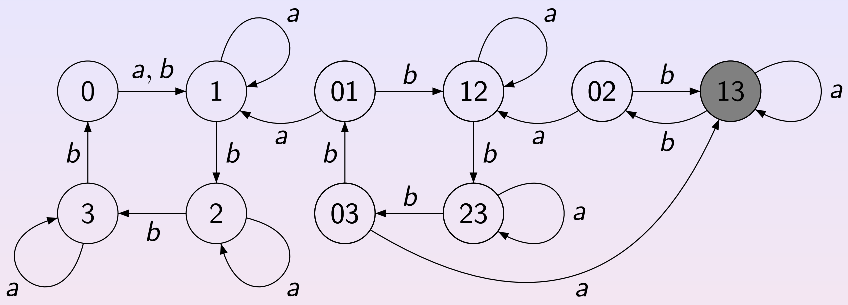
6. Example



$$a, \;\; Q \,.\, a = \{1, 2, 3\}; \qquad a \cdot bba, \;\; Q \,.\, abba = \{1, 3\}$$

$$abba \cdot babbba, \;\; Q \,.\, abbababbba = \{1\}$$

Observe that the reset word constructed this way is of length 10 while we know a reset word of length 9.

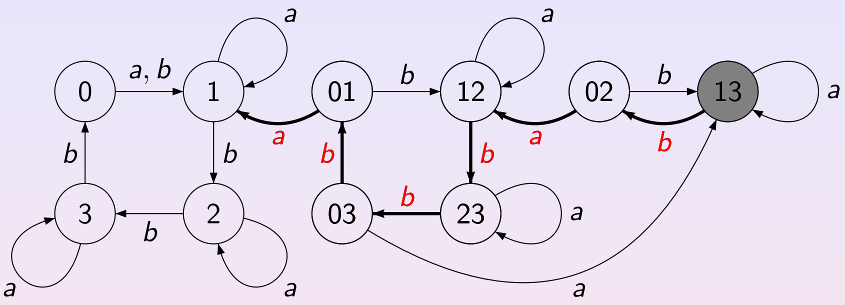


$$a, \;\; Q\,.\,a = \{1,2,3\}; \qquad a \cdot bba, \;\; Q\,.\,abba = \{1,3\}$$

$$abba \cdot babbba, \;\; Q\,.\,abbababbba = \{1\}$$

Observe that the reset word constructed this way is of length 10 while we know a reset word of length 9.

6. Example

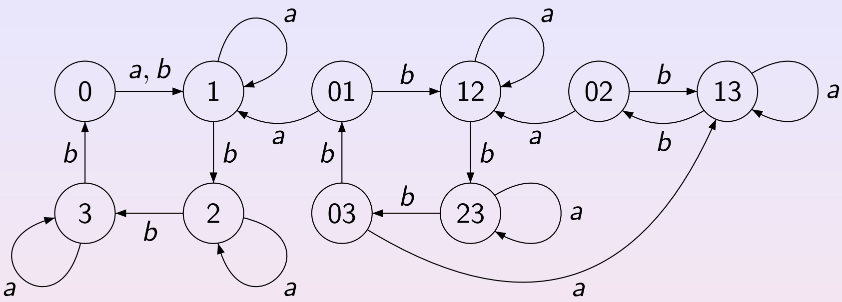


$$a, \;\; Q\,.\,a = \{1,2,3\}; \qquad a \cdot bba, \;\; Q\,.\,abba = \{1,3\}$$

$$abba \cdot babbba, \;\; Q\,.\,abbababbba = \{1\}$$

Observe that the reset word constructed this way is of length 10 while we know a reset word of length 9.

6. Example



$$a,\ Q\,.\,a = \{1,2,3\};\quad a \cdot bba,\ Q\,.\,abba = \{1,3\}$$

$$abba \cdot babbba,\ Q\,.\,abbababbba = \{1\}$$

Observe that the reset word constructed this way is of length 10 while we know a reset word of length 9.

7. Results-I

Thus, recognizing synchronizability reduces to a reachability problem in the automaton whose states are the 2-subsets and the 1-subsets of Q. The latter can be solved by BFS in $O(n^2 \cdot |\Sigma|)$ time where $n = |Q|$.

Can one do better? It is an open problem.

Recently, Mikhail Berlinkov has developed a (non-trivial) algorithm that checks whether or not an automaton with n states is synchronizing and spends time $O(n)$ on average. The worst case complexity of Berlinkov's algorithm is still quadratic.

Thus, recognizing synchronizability reduces to a reachability problem in the automaton whose states are the 2-subsets and the 1-subsets of Q. The latter can be solved by BFS in $O(n^2 \cdot |\Sigma|)$ time where $n = |Q|$.

Can one do better? It is an open problem.

Recently, Mikhail Berlinkov has developed a (non-trivial) algorithm that checks whether or not an automaton with n states is synchronizing and spends time $O(n)$ on average. The worst case complexity of Berlinkov's algorithm is still quadratic.

Thus, recognizing synchronizability reduces to a reachability problem in the automaton whose states are the 2-subsets and the 1-subsets of Q. The latter can be solved by BFS in $O(n^2 \cdot |\Sigma|)$ time where $n = |Q|$.

Can one do better? It is an open problem.

Recently, Mikhail Berlinkov has developed a (non-trivial) algorithm that checks whether or not an automaton with n states is synchronizing and spends time $O(n)$ on average. The worst case complexity of Berlinkov's algorithm is still quadratic.

7. Results-I

Thus, recognizing synchronizability reduces to a reachability problem in the automaton whose states are the 2-subsets and the 1-subsets of Q. The latter can be solved by BFS in $O(n^2 \cdot |\Sigma|)$ time where $n = |Q|$.

Can one do better? It is an open problem.

Recently, Mikhail Berlinkov has developed a (non-trivial) algorithm that checks whether or not an automaton with n states is synchronizing and spends time $O(n)$ on average. The worst case complexity of Berlinkov's algorithm is still quadratic.

7. Results-I

Thus, recognizing synchronizability reduces to a reachability problem in the automaton whose states are the 2-subsets and the 1-subsets of Q. The latter can be solved by BFS in $O(n^2 \cdot |\Sigma|)$ time where $n = |Q|$.

Can one do better? It is an open problem.

Recently, Mikhail Berlinkov has developed a (non-trivial) algorithm that checks whether or not an automaton with n states is synchronizing and spends time $O(n)$ on average. The worst case complexity of Berlinkov's algorithm is still quadratic.

8. Results-II

In fact, the basic algorithm not only recognizes synchronizability but also returns a reset word provided that such exists.

If one also wants to produce a reset word, one need $O(n^3 + n^2 \cdot |\Sigma|)$ time. Why? One needs time to write down the word!

Clearly, the resulting reset word has length $O(n^3)$: the algorithm makes at most $n-1$ steps and the length of the segment added in the step when k states are still to be compressed ($n \geq k \geq 2$) is at most $1 + \#$ of blank 2-subsets, i.e., $1 + \binom{n}{2} - \binom{k}{2}$. This gives the upper bound close to $\frac{n^3 - n}{3}$.

Can we do better? What is the exact bound?

8. Results-II

In fact, the basic algorithm not only recognizes synchronizability but also returns a reset word provided that such exists.
If one also wants to produce a reset word, one need $O(n^3 + n^2 \cdot |\Sigma|)$ time. Why? One needs time to write down the word!

Clearly, the resulting reset word has length $O(n^3)$: the algorithm makes at most $n-1$ steps and the length of the segment added in the step when k states are still to be compressed $(n \geq k \geq 2)$ is at most $1 + \#$ of blank 2-subsets, i.e., $1 + \binom{n}{2} - \binom{k}{2}$. This gives the upper bound close to $\frac{n^3-n}{3}$.

Can we do better? What is the exact bound?

In fact, the basic algorithm not only recognizes synchronizability but also returns a reset word provided that such exists.
If one also wants to produce a reset word, one need $O(n^3 + n^2 \cdot |\Sigma|)$ time. Why? One needs time to write down the word!

Clearly, the resulting reset word has length $O(n^3)$: the algorithm makes at most $n-1$ steps and the length of the segment added in the step when k states are still to be compressed ($n \geq k \geq 2$) is at most $1 + \#$ of blank 2-subsets, i.e., $1 + \binom{n}{2} - \binom{k}{2}$. This gives the upper bound close to $\frac{n^3-n}{3}$.

Can we do better? What is the exact bound?

8. Results-II

In fact, the basic algorithm not only recognizes synchronizability but also returns a reset word provided that such exists.
If one also wants to produce a reset word, one need $O(n^3 + n^2 \cdot |\Sigma|)$ time. Why? One needs time to write down the word!

Clearly, the resulting reset word has length $O(n^3)$: the algorithm makes at most $n-1$ steps and the length of the segment added in the step when k states are still to be compressed $(n \geq k \geq 2)$ is at most $1 + \#$ of blank 2-subsets, i.e., $1 + \binom{n}{2} - \binom{k}{2}$. This gives the upper bound close to $\frac{n^3-n}{3}$.

Can we do better? What is the exact bound?

8. Results-II

In fact, the basic algorithm not only recognizes synchronizability but also returns a reset word provided that such exists.
If one also wants to produce a reset word, one need $O(n^3 + n^2 \cdot |\Sigma|)$ time. Why? One needs time to write down the word!

Clearly, the resulting reset word has length $O(n^3)$: the algorithm makes at most $n - 1$ steps and the length of the segment added in the step when k states are still to be compressed $(n \geq k \geq 2)$ is at most $1 + \#$ of blank 2-subsets, i.e., $1 + \binom{n}{2} - \binom{k}{2}$. This gives the upper bound close to $\frac{n^3-n}{3}$.

Can we do better? What is the exact bound?

8. Results-II

In fact, the basic algorithm not only recognizes synchronizability but also returns a reset word provided that such exists.
If one also wants to produce a reset word, one need $O(n^3 + n^2 \cdot |\Sigma|)$ time. Why? One needs time to write down the word!

Clearly, the resulting reset word has length $O(n^3)$: the algorithm makes at most $n - 1$ steps and the length of the segment added in the step when k states are still to be compressed $(n \geq k \geq 2)$ is at most $1 + \#$ of blank 2-subsets, i.e., $1 + \binom{n}{2} - \binom{k}{2}$. This gives the upper bound close to $\frac{n^3-n}{3}$.

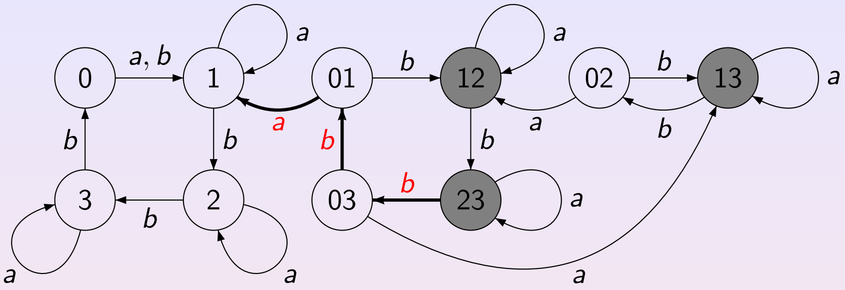
Can we do better? What is the exact bound?

8. Results-II

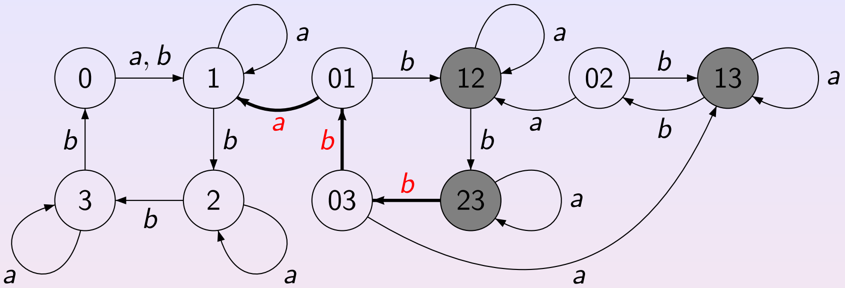
In fact, the basic algorithm not only recognizes synchronizability but also returns a reset word provided that such exists.
If one also wants to produce a reset word, one need $O(n^3 + n^2 \cdot |\Sigma|)$ time. Why? One needs time to write down the word!

Clearly, the resulting reset word has length $O(n^3)$: the algorithm makes at most $n-1$ steps and the length of the segment added in the step when k states are still to be compressed $(n \geq k \geq 2)$ is at most $1 + \#$ of blank 2-subsets, i.e., $1 + \binom{n}{2} - \binom{k}{2}$. This gives the upper bound close to $\frac{n^3-n}{3}$.
Can we do better? What is the exact bound?

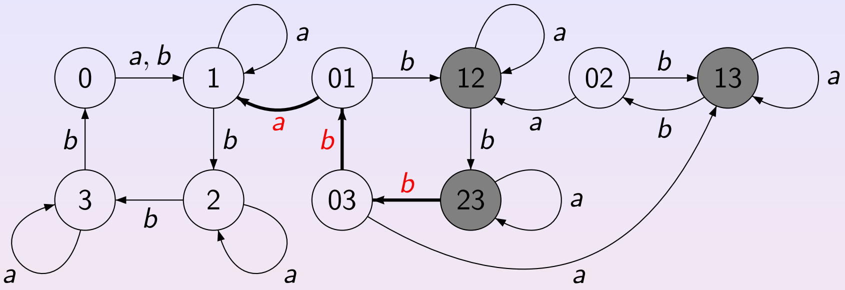
9. A Resource for Improvement



We see that the shortest path from a light-grey 2-subset to a singleton does not necessarily pass through all blank 2-subsets. Consider a generic step of the algorithm at which states to be compressed form a set P with $|P| = k > 1$ and let $v = a_1 \cdots a_\ell$ with $a_i \in \Sigma$, $i = 1, \ldots, \ell$, be a word of minimum length such that $|P\,.\,v| < k$.

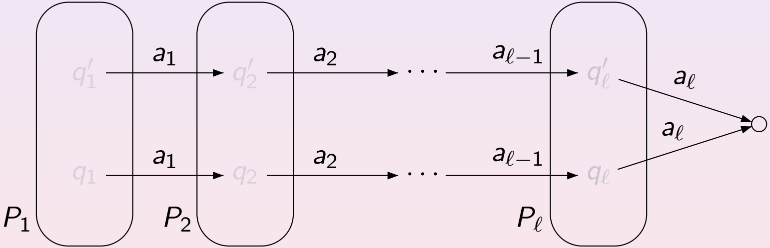


We see that the shortest path from a light-grey 2-subset to a singleton does not necessarily pass through all blank 2-subsets.
Consider a generic step of the algorithm at which states to be compressed form a set P with $|P| = k > 1$ and let $v = a_1 \cdots a_\ell$ with $a_i \in \Sigma$, $i = 1, \ldots, \ell$, be a word of minimum length such that $|P\,.\,v| < k$.



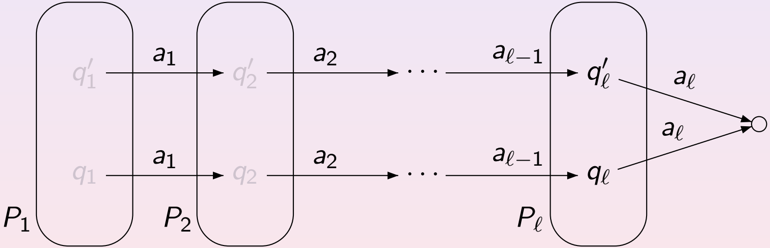
We see that the shortest path from a light-grey 2-subset to a singleton does not necessarily pass through all blank 2-subsets. Consider a generic step of the algorithm at which states to be compressed form a set P with $|P| = k > 1$ and let $v = a_1 \cdots a_\ell$ with $a_i \in \Sigma$, $i = 1, \ldots, \ell$, be a word of minimum length such that $|P \,.\, v| < k$.

The sets $P_1 = P$, $P_2 = P_1 . a_1$, …, $P_\ell = P_{\ell-1} . a_{\ell-1}$ are k-subsets of Q. Since $|P_\ell . a_\ell| < |P_\ell|$, there exist two states $q_\ell, q'_\ell \in P_\ell$ such that $\delta(q_\ell, a_\ell) = \delta(q'_\ell, a_\ell)$. Now define 2-subsets $R_i = \{q_i, q'_i\} \subseteq P_i$, $i = 1, \ldots, \ell$, such that $\delta(q_i, a_i) = q_{i+1}$, $\delta(q'_i, a_i) = q'_{i+1}$ for $i = 1, \ldots, \ell - 1$.



The condition that v is a word of minimum length with $|P . v| < |P|$ implies $R_i \not\subseteq P_j$ for $1 \le j < i \le \ell$.

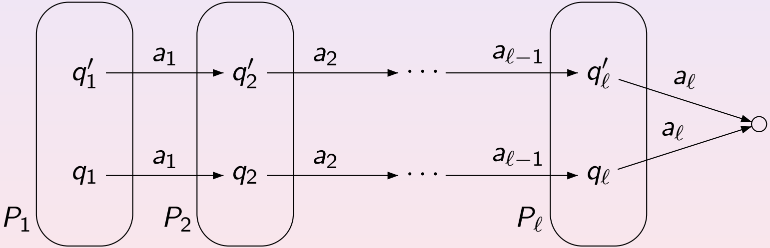
The sets $P_1 = P$, $P_2 = P_1 \,.\, a_1$, $\ldots$, $P_\ell = P_{\ell-1} \,.\, a_{\ell-1}$ are k-subsets of Q. Since $|P_\ell \,.\, a_\ell| < |P_\ell|$, there exist two states $q_\ell, q'_\ell \in P_\ell$ such that $\delta(q_\ell, a_\ell) = \delta(q'_\ell, a_\ell)$. Now define 2-subsets $R_i = \{q_i, q'_i\} \subseteq P_i$, $i = 1, \ldots, \ell$, such that $\delta(q_i, a_i) = q_{i+1}$, $\delta(q'_i, a_i) = q'_{i+1}$ for $i = 1, \ldots, \ell - 1$.



The condition that v is a word of minimum length with $|P \,.\, v| < |P|$ implies $R_i \not\subseteq P_j$ for $1 \leq j < i \leq \ell$.

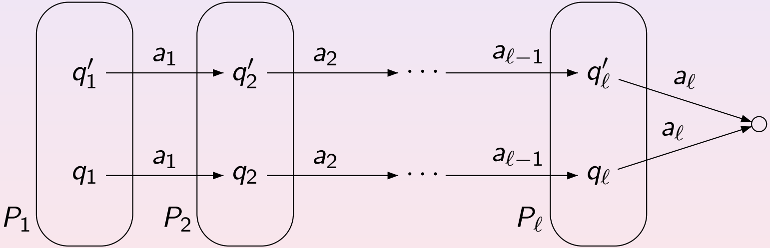
10. Studying Generic Step

The sets $P_1 = P$, $P_2 = P_1 . a_1$, $\dots$, $P_\ell = P_{\ell-1} . a_{\ell-1}$ are k-subsets of Q. Since $|P_\ell . a_\ell| < |P_\ell|$, there exist two states $q_\ell, q'_\ell \in P_\ell$ such that $\delta(q_\ell, a_\ell) = \delta(q'_\ell, a_\ell)$. Now define 2-subsets $R_i = \{q_i, q'_i\} \subseteq P_i$, $i = 1, \dots, \ell$, such that $\delta(q_i, a_i) = q_{i+1}$, $\delta(q'_i, a_i) = q'_{i+1}$ for $i = 1, \dots, \ell - 1$.



The condition that v is a word of minimum length with $|P . v| < |P|$ implies $R_i \not\subseteq P_j$ for $1 \le j < i \le \ell$.

The sets $P_1 = P,\ P_2 = P_1 \,.\, a_1,\ \dots,\ P_\ell = P_{\ell-1} \,.\, a_{\ell-1}$ are k-subsets of Q. Since $|P_\ell \,.\, a_\ell| < |P_\ell|$, there exist two states $q_\ell, q'_\ell \in P_\ell$ such that $\delta(q_\ell, a_\ell) = \delta(q'_\ell, a_\ell)$. Now define 2-subsets $R_i = \{q_i, q'_i\} \subseteq P_i$, $i = 1, \dots, \ell$, such that $\delta(q_i, a_i) = q_{i+1}$, $\delta(q'_i, a_i) = q'_{i+1}$ for $i = 1, \dots, \ell - 1$.



The condition that v is a word of minimum length with $|P \,.\, v| < |P|$ implies $R_i \nsubseteq P_j$ for $1 \le j < i \le \ell$.

11. Combinatorial Configuration

Our question reduces to the following problem in combinatorics of finite sets:

Let Q be an n-set, $P_1, \ldots, P_\ell$ a sequence of its k-subsets ($k > 1$) such that each P_i, $1 < i \leq \ell$, includes a "fresh" 2-subset that does not occur in any previous P_j ($1 \leq j < i$). How long can such renewing sequences be?

A construction: fix a $(k-2)$-subset W of Q, list all $\binom{n-k+2}{2}$ 2-subsets of $Q \setminus W$ and let T_i be the union of W with the i^{th} 2-subset in the list. This gives the renewing sequence $T_1, \ldots, T_s$ of length $s = \binom{n-k+2}{2}$. Is this the maximum?

Our question reduces to the following problem in combinatorics of finite sets:

Let Q be an n-set, $P_1, \dots, P_\ell$ a sequence of its k-subsets ($k > 1$) such that each P_i, $1 < i \leq \ell$, includes a "fresh" 2-subset that does not occur in any previous P_j ($1 \leq j < i$). How long can such renewing sequences be?

A construction: fix a $(k-2)$-subset W of Q, list all $\binom{n-k+2}{2}$ 2-subsets of $Q \setminus W$ and let T_i be the union of W with the i^{th} 2-subset in the list. This gives the renewing sequence $T_1, \dots, T_s$ of length $s = \binom{n-k+2}{2}$. Is this the maximum?

11. Combinatorial Configuration

Our question reduces to the following problem in combinatorics of finite sets:

Let Q be an n-set, $P_1, \dots, P_\ell$ a sequence of its k-subsets ($k > 1$) such that each P_i, $1 < i \leq \ell$, includes a "fresh" 2-subset that does not occur in any previous P_j $(1 \leq j < i)$. How long can such renewing sequences be?

A construction: fix a $(k-2)$-subset W of Q, list all $\binom{n-k+2}{2}$ 2-subsets of $Q \setminus W$ and let T_i be the union of W with the i^{th} 2-subset in the list. This gives the renewing sequence $T_1, \dots, T_s$ of length $s = \binom{n-k+2}{2}$. Is this the maximum?

Our question reduces to the following problem in combinatorics of finite sets:

Let Q be an n-set, $P_1, \ldots, P_\ell$ a sequence of its k-subsets ($k > 1$) such that each P_i, $1 < i \leq \ell$, includes a "fresh" 2-subset that does not occur in any previous P_j ($1 \leq j < i$). How long can such renewing sequences be?

A construction: fix a $(k-2)$-subset W of Q, list all $\binom{n-k+2}{2}$ 2-subsets of $Q \setminus W$ and let T_i be the union of W with the i^{th} 2-subset in the list. This gives the renewing sequence $T_1, \ldots, T_s$ of length $s = \binom{n-k+2}{2}$. Is this the maximum?

Our question reduces to the following problem in combinatorics of finite sets:

Let Q be an n-set, $P_1, \dots, P_\ell$ a sequence of its k-subsets ($k > 1$) such that each P_i, $1 < i \leq \ell$, includes a "fresh" 2-subset that does not occur in any previous P_j ($1 \leq j < i$). How long can such renewing sequences be?

A construction: fix a $(k-2)$-subset W of Q, list all $\binom{n-k+2}{2}$ 2-subsets of $Q \setminus W$ and let T_i be the union of W with the i^{th} 2-subset in the list. This gives the renewing sequence $T_1, \dots, T_s$ of length $s = \binom{n-k+2}{2}$. Is this the maximum?

Our question reduces to the following problem in combinatorics of finite sets:

Let Q be an n-set, $P_1, \dots, P_\ell$ a sequence of its k-subsets ($k > 1$) such that each P_i, $1 < i \le \ell$, includes a "fresh" 2-subset that does not occur in any previous P_j ($1 \le j < i$). How long can such renewing sequences be?

A construction: fix a $(k-2)$-subset W of Q, list all $\binom{n-k+2}{2}$ 2-subsets of $Q \setminus W$ and let T_i be the union of W with the i^{th} 2-subset in the list. This gives the renewing sequence $T_1, \dots, T_s$ of length $s = \binom{n-k+2}{2}$. Is this the maximum?

The question turned out to be very difficult and was solved (in the affirmative) by Peter Frankl (An extremal problem for two families of sets, Eur. J. Comb., 3 (1982) 125–127).

The proof uses linearization techniques which are quite common in combinatorics of finite sets. One reformulates the problem in linear algebra terms and then uses the corresponding machinery.

We identify Q with $\{1, 2, \ldots, n\}$ and assign to each k-subset $I = \{i_1, \ldots, i_k\}$ the following polynomial $D(I)$ in variables $x_{i_1}, \ldots, x_{i_k}$ over the field of reals.

The question turned out to be very difficult and was solved (in the affirmative) by Peter Frankl (An extremal problem for two families of sets, Eur. J. Comb., 3 (1982) 125–127).

The proof uses linearization techniques which are quite common in combinatorics of finite sets. One reformulates the problem in linear algebra terms and then uses the corresponding machinery.

We identify Q with $\{1, 2, \ldots, n\}$ and assign to each k-subset $I = \{i_1, \ldots, i_k\}$ the following polynomial $D(I)$ in variables $x_{i_1}, \ldots, x_{i_k}$ over the field of reals.

The question turned out to be very difficult and was solved (in the affirmative) by Peter Frankl (An extremal problem for two families of sets, Eur. J. Comb., 3 (1982) 125–127).

The proof uses linearization techniques which are quite common in combinatorics of finite sets. One reformulates the problem in linear algebra terms and then uses the corresponding machinery.

We identify Q with $\{1, 2, \dots, n\}$ and assign to each k-subset $I = \{i_1, \dots, i_k\}$ the following polynomial $D(I)$ in variables $x_{i_1}, \dots, x_{i_k}$ over the field of reals.

The question turned out to be very difficult and was solved (in the affirmative) by Peter Frankl (An extremal problem for two families of sets, Eur. J. Comb., 3 (1982) 125–127).

The proof uses linearization techniques which are quite common in combinatorics of finite sets. One reformulates the problem in linear algebra terms and then uses the corresponding machinery.

We identify Q with $\{1, 2, \ldots, n\}$ and assign to each k-subset $I = \{i_1, \ldots, i_k\}$ the following polynomial $D(I)$ in variables $x_{i_1}, \ldots, x_{i_k}$ over the field of reals.

13. Linearization

$$I = \{i_1, \ldots, i_k\} \mapsto D(I) = \begin{vmatrix} 1 & i_1 & i_1^2 & \cdots & i_1^{k-3} & x_{i_1} & x_{i_1}^2 \\ 1 & i_2 & i_2^2 & \cdots & i_2^{k-3} & x_{i_2} & x_{i_2}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 1 & i_k & i_k^2 & \cdots & i_k^{k-3} & x_{i_k} & x_{i_k}^2 \end{vmatrix}_{k\times k}$$

Then one proves that:

• the polynomials $D(P_1), \ldots, D(P_\ell)$ are linearly independent whenever the k-subsets $P_1, \ldots, P_\ell$ form a renewing sequence;

• the polynomials $D(T_1), \ldots, D(T_s)$ (derived from the "standard" sequence) generate the linear space spanned by all polynomials of the form $D(I)$.

13. Linearization

$$I = \{i_1, \ldots, i_k\} \mapsto D(I) = \begin{vmatrix} 1 & i_1 & i_1^2 & \cdots & i_1^{k-3} & x_{i_1} & x_{i_1}^2 \\ 1 & i_2 & i_2^2 & \cdots & i_2^{k-3} & x_{i_2} & x_{i_2}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 1 & i_k & i_k^2 & \cdots & i_k^{k-3} & x_{i_k} & x_{i_k}^2 \end{vmatrix}_{k\times k}$$

Then one proves that:

• the polynomials $D(P_1), \ldots, D(P_\ell)$ are linearly independent whenever the k-subsets $P_1, \ldots, P_\ell$ form a renewing sequence;

• the polynomials $D(T_1), \ldots, D(T_s)$ (derived from the "standard" sequence) generate the linear space spanned by all polynomials of the form $D(I)$.

13. Linearization

$$I = \{i_1, \dots, i_k\} \mapsto D(I) = \begin{vmatrix} 1 & i_1 & i_1^2 & \cdots & i_1^{k-3} & x_{i_1} & x_{i_1}^2 \\ 1 & i_2 & i_2^2 & \cdots & i_2^{k-3} & x_{i_2} & x_{i_2}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 1 & i_k & i_k^2 & \cdots & i_k^{k-3} & x_{i_k} & x_{i_k}^2 \end{vmatrix}_{k\times k}$$

Then one proves that:
• the polynomials $D(P_1), \dots, D(P_\ell)$ are linearly independent whenever the k-subsets $P_1, \dots, P_\ell$ form a renewing sequence;
• the polynomials $D(T_1), \dots, D(T_s)$ (derived from the "standard" sequence) generate the linear space spanned by all polynomials of the form $D(I)$.

14. Linearization, Step 1

$$D(I) = \begin{vmatrix} 1 & i_1 & i_1^2 & \cdots & i_1^{k-3} & x_{i_1} & x_{i_1}^2 \\ 1 & i_2 & i_2^2 & \cdots & i_2^{k-3} & x_{i_2} & x_{i_2}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 1 & i_k & i_k^2 & \cdots & i_k^{k-3} & x_{i_k} & x_{i_k}^2 \end{vmatrix}_{k\times k}$$

Suppose that k-subsets $P_1, \dots, P_\ell$ form a renewing sequence but $D(P_1), \dots, D(P_\ell)$ are linearly dependent. Then some polynomial $D(P_j)$ should be expressible as a linear combination of the preceding polynomials $D(P_1), \dots, D(P_{j-1})$. By the definition of a renewing sequence, P_j contains a couple $\{p, p'\}$ such that $\{p, p'\} \not\subseteq P_i$ for all $i < j$. If we substitute $x_p = p$, $x_{p'} = p'$ and $x_t = 0$ for $t \neq p, p'$ in each polynomial $D(P_1), \dots, D(P_j)$, then the polynomials $D(P_1), \dots, D(P_{j-1})$ vanish (since the two last columns in each of the resulting determinants become proportional) and so does any linear combination of the polynomials.

14. Linearization, Step 1

$$D(I) = \begin{vmatrix} 1 & i_1 & i_1^2 & \cdots & i_1^{k-3} & x_{i_1} & x_{i_1}^2 \\ 1 & i_2 & i_2^2 & \cdots & i_2^{k-3} & x_{i_2} & x_{i_2}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 1 & i_k & i_k^2 & \cdots & i_k^{k-3} & x_{i_k} & x_{i_k}^2 \end{vmatrix}_{k \times k}$$

Suppose that k-subsets $P_1, \ldots, P_\ell$ form a renewing sequence but $D(P_1), \ldots, D(P_\ell)$ are linearly dependent. Then some polynomial $D(P_j)$ should be expressible as a linear combination of the preceding polynomials $D(P_1), \ldots, D(P_{j-1})$. By the definition of a renewing sequence, P_j contains a couple $\{p, p'\}$ such that $\{p, p'\} \nsubseteq P_i$ for all $i < j$. If we substitute $x_p = p$, $x_{p'} = p'$ and $x_t = 0$ for $t \neq p, p'$ in each polynomial $D(P_1), \ldots, D(P_j)$, then the polynomials $D(P_1), \ldots, D(P_{j-1})$ vanish (since the two last columns in each of the resulting determinants become proportional) and so does any linear combination of the polynomials.

14. Linearization, Step 1

$$D(I) = \begin{vmatrix} 1 & i_1 & i_1^2 & \cdots & i_1^{k-3} & x_{i_1} & x_{i_1}^2 \\ 1 & i_2 & i_2^2 & \cdots & i_2^{k-3} & x_{i_2} & x_{i_2}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 1 & i_k & i_k^2 & \cdots & i_k^{k-3} & x_{i_k} & x_{i_k}^2 \end{vmatrix}_{k \times k}$$

Suppose that k-subsets $P_1, \ldots, P_\ell$ form a renewing sequence but $D(P_1), \ldots, D(P_\ell)$ are linearly dependent. Then some polynomial $D(P_j)$ should be expressible as a linear combination of the preceding polynomials $D(P_1), \ldots, D(P_{j-1})$. By the definition of a renewing sequence, P_j contains a couple $\{p, p'\}$ such that $\{p, p'\} \nsubseteq P_i$ for all $i < j$. If we substitute $x_p = p$, $x_{p'} = p'$ and $x_t = 0$ for $t \neq p, p'$ in each polynomial $D(P_1), \ldots, D(P_j)$, then the polynomials $D(P_1), \ldots, D(P_{j-1})$ vanish (since the two last columns in each of the resulting determinants become proportional) and so does any linear combination of the polynomials.

14. Linearization, Step 1

$$D(I) = \begin{vmatrix} 1 & i_1 & i_1^2 & \cdots & i_1^{k-3} & x_{i_1} & x_{i_1}^2 \\ 1 & i_2 & i_2^2 & \cdots & i_2^{k-3} & x_{i_2} & x_{i_2}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 1 & i_k & i_k^2 & \cdots & i_k^{k-3} & x_{i_k} & x_{i_k}^2 \end{vmatrix}_{k\times k}$$

Suppose that k-subsets $P_1, \dots, P_\ell$ form a renewing sequence but $D(P_1), \dots, D(P_\ell)$ are linearly dependent. Then some polynomial $D(P_j)$ should be expressible as a linear combination of the preceding polynomials $D(P_1), \dots, D(P_{j-1})$. By the definition of a renewing sequence, P_j contains a couple $\{p, p'\}$ such that $\{p, p'\} \not\subseteq P_i$ for all $i < j$. If we substitute $x_p = p$, $x_{p'} = p'$ and $x_t = 0$ for $t \neq p, p'$ in each polynomial $D(P_1), \dots, D(P_j)$, then the polynomials $D(P_1), \dots, D(P_{j-1})$ vanish (since the two last columns in each of the resulting determinants become proportional) and so does any linear combination of the polynomials.

14. Linearization, Step 1

$$D(I) = \begin{vmatrix} 1 & i_1 & i_1^2 & \cdots & i_1^{k-3} & x_{i_1} & x_{i_1}^2 \\ 1 & i_2 & i_2^2 & \cdots & i_2^{k-3} & x_{i_2} & x_{i_2}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 1 & i_k & i_k^2 & \cdots & i_k^{k-3} & x_{i_k} & x_{i_k}^2 \end{vmatrix}_{k \times k}$$

Suppose that k-subsets $P_1, \dots, P_\ell$ form a renewing sequence but $D(P_1), \dots, D(P_\ell)$ are linearly dependent. Then some polynomial $D(P_j)$ should be expressible as a linear combination of the preceding polynomials $D(P_1), \dots, D(P_{j-1})$. By the definition of a renewing sequence, P_j contains a couple $\{p, p'\}$ such that $\{p, p'\} \nsubseteq P_i$ for all $i < j$. If we substitute $x_p = p$, $x_{p'} = p'$ and $x_t = 0$ for $t \neq p, p'$ in each polynomial $D(P_1), \dots, D(P_j)$, then the polynomials $D(P_1), \dots, D(P_{j-1})$ vanish (since the two last columns in each of the resulting determinants become proportional) and so does any linear combination of the polynomials.

15. Linearization, Step 1, completed

$$D(P_j) = \begin{vmatrix} 1 & i_1 & i_1^2 & \cdots & i_1^{k-3} & p & p^2 \\ 1 & i_2 & i_2^2 & \cdots & i_2^{k-3} & p' & (p')^2 \\ 1 & i_3 & i_3^2 & \cdots & i_3^{k-3} & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 1 & i_k & i_k^2 & \cdots & i_k^{k-3} & 0 & 0 \end{vmatrix}_{k\times k}$$

(For simplicity, here we assume that $i_1 = p$, $i_2 = p'$.)

The value of $D(P_j)$ is the determinant being the product of a Vandermonde $(k-2)\times(k-2)$-determinant with the 2×2-determinant $\begin{vmatrix} p & p^2 \\ p' & (p')^2 \end{vmatrix}$, whence this value is not 0.

Hence $D(P_j)$ cannot be equal to a linear combination of $D(P_1), \ldots, D(P_{j-1})$.

15. Linearization, Step 1, completed

$$D(P_j) = \begin{vmatrix} 1 & i_1 & i_1^2 & \cdots & i_1^{k-3} & p & p^2 \\ 1 & i_2 & i_2^2 & \cdots & i_2^{k-3} & p' & (p')^2 \\ 1 & i_3 & i_3^2 & \cdots & i_3^{k-3} & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 1 & i_k & i_k^2 & \cdots & i_k^{k-3} & 0 & 0 \end{vmatrix}_{k\times k}$$

(For simplicity, here we assume that $i_1 = p$, $i_2 = p'$.)

The value of $D(P_j)$ is the determinant being the product of a Vandermonde $(k-2)\times(k-2)$-determinant with the 2×2-determinant $\begin{vmatrix} p & p^2 \\ p' & (p')^2 \end{vmatrix}$, whence this value is not 0.

Hence $D(P_j)$ cannot be equal to a linear combination of $D(P_1), \ldots, D(P_{j-1})$.

16. Linearization, Step 2

Now we aim to prove that the polynomials $D(T_1), \dots, D(T_s)$ (derived from the "standard" sequence) generate the linear space spanned by all polynomials of the form $D(I)$. Take an arbitrary k-element subset $I = \{i_1, \dots, i_k\}$ of Q. We claim that the polynomial $D(I)$ is a linear combination of $D(T_1), \dots, D(T_s)$.

We induct on the cardinality of the set $I \setminus W$. If $|I \setminus W| = 2$, then I is the union of W with some couple from $Q \setminus W$, whence $I = T_i$ for some $i = 1, \dots, s$. Thus, $D(I) = D(T_i)$ and our claim holds true.

16. Linearization, Step 2

Now we aim to prove that the polynomials $D(T_1), \dots, D(T_s)$ (derived from the "standard" sequence) generate the linear space spanned by all polynomials of the form $D(I)$. Take an arbitrary k-element subset $I = \{i_1, \dots, i_k\}$ of Q. We claim that the polynomial $D(I)$ is a linear combination of $D(T_1), \dots, D(T_s)$.

We induct on the cardinality of the set $I \setminus W$. If $|I \setminus W| = 2$, then I is the union of W with some couple from $Q \setminus W$, whence $I = T_i$ for some $i = 1, \dots, s$. Thus, $D(I) = D(T_i)$ and our claim holds true.

16. Linearization, Step 2

Now we aim to prove that the polynomials $D(T_1), \ldots, D(T_s)$ (derived from the “standard” sequence) generate the linear space spanned by all polynomials of the form $D(I)$. Take an arbitrary k-element subset $I = \{i_1, \ldots, i_k\}$ of Q. We claim that the polynomial $D(I)$ is a linear combination of $D(T_1), \ldots, D(T_s)$.

We induct on the cardinality of the set $I \setminus W$. If $|I \setminus W| = 2$, then I is the union of W with some couple from $Q \setminus W$, whence $I = T_i$ for some $i = 1, \ldots, s$. Thus, $D(I) = D(T_i)$ and our claim holds true.

Now we aim to prove that the polynomials $D(T_1), \ldots, D(T_s)$ (derived from the "standard" sequence) generate the linear space spanned by all polynomials of the form $D(I)$. Take an arbitrary k-element subset $I = \{i_1, \ldots, i_k\}$ of Q. We claim that the polynomial $D(I)$ is a linear combination of $D(T_1), \ldots, D(T_s)$.

We induct on the cardinality of the set $I \setminus W$. If $|I \setminus W| = 2$, then I is the union of W with some couple from $Q \setminus W$, whence $I = T_i$ for some $i = 1, \ldots, s$. Thus, $D(I) = D(T_i)$ and our claim holds true.

17. Linearization, Step 2, continued

If $|I \setminus W| > 2$, there is $i_0 \in W \setminus I$. Let $I' = I \cup \{i_0\}$. There exists a polynomial $p(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 \cdots + \alpha_{k-3} x^{k-3}$ over $\mathbb{R}$ such that $p(i_0) = 1$ and $p(i) = 0$ for all $i \in W \setminus \{i_0\}$. Consider the determinant

$$\Delta = \begin{vmatrix} p(i_0) & 1 & i_0 & i_0^2 & \cdots & i_0^{k-3} & x_{i_0} & x_{i_0}^2 \\ p(i_1) & 1 & i_1 & i_1^2 & \cdots & i_1^{k-3} & x_{i_1} & x_{i_1}^2 \\ p(i_2) & 1 & i_2 & i_2^2 & \cdots & i_2^{k-3} & x_{i_2} & x_{i_2}^2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ p(i_k) & 1 & i_k & i_k^2 & \cdots & i_k^{k-3} & x_{i_k} & x_{i_k}^2 \end{vmatrix}_{(k+1)\times(k+1)}.$$

Clearly, $\Delta = 0$ as the first column is the sum of the next $k-2$ columns with the coefficients $\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_{k-3}$.

17. Linearization, Step 2, continued

If $|I \setminus W| > 2$, there is $i_0 \in W \setminus I$. Let $I' = I \cup \{i_0\}$. There exists a polynomial $p(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 \cdots + \alpha_{k-3} x^{k-3}$ over $\mathbb{R}$ such that $p(i_0) = 1$ and $p(i) = 0$ for all $i \in W \setminus \{i_0\}$. Consider the determinant

$$\Delta = \begin{vmatrix} p(i_0) & 1 & i_0 & i_0^2 & \cdots & i_0^{k-3} & x_{i_0} & x_{i_0}^2 \\ p(i_1) & 1 & i_1 & i_1^2 & \cdots & i_1^{k-3} & x_{i_1} & x_{i_1}^2 \\ p(i_2) & 1 & i_2 & i_2^2 & \cdots & i_2^{k-3} & x_{i_2} & x_{i_2}^2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ p(i_k) & 1 & i_k & i_k^2 & \cdots & i_k^{k-3} & x_{i_k} & x_{i_k}^2 \end{vmatrix}_{(k+1)\times(k+1)}.$$

Clearly, $\Delta = 0$ as the first column is the sum of the next $k-2$ columns with the coefficients $\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_{k-3}$.

17. Linearization, Step 2, continued

If $|I \setminus W| > 2$, there is $i_0 \in W \setminus I$. Let $I' = I \cup \{i_0\}$. There exists a polynomial $p(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 \cdots + \alpha_{k-3} x^{k-3}$ over $\mathbb{R}$ such that $p(i_0) = 1$ and $p(i) = 0$ for all $i \in W \setminus \{i_0\}$. Consider the determinant

$$\Delta = \begin{vmatrix} p(i_0) & 1 & i_0 & i_0^2 & \cdots & i_0^{k-3} & x_{i_0} & x_{i_0}^2 \\ p(i_1) & 1 & i_1 & i_1^2 & \cdots & i_1^{k-3} & x_{i_1} & x_{i_1}^2 \\ p(i_2) & 1 & i_2 & i_2^2 & \cdots & i_2^{k-3} & x_{i_2} & x_{i_2}^2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ p(i_k) & 1 & i_k & i_k^2 & \cdots & i_k^{k-3} & x_{i_k} & x_{i_k}^2 \end{vmatrix}_{(k+1)\times(k+1)}.$$

Clearly, $\Delta = 0$ as the first column is the sum of the next $k-2$ columns with the coefficients $\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_{k-3}$.

17. Linearization, Step 2, continued

If $|I \setminus W| > 2$, there is $i_0 \in W \setminus I$. Let $I' = I \cup \{i_0\}$. There exists a polynomial $p(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 \cdots + \alpha_{k-3} x^{k-3}$ over $\mathbb{R}$ such that $p(i_0) = 1$ and $p(i) = 0$ for all $i \in W \setminus \{i_0\}$. Consider the determinant

$$\Delta = \begin{vmatrix} p(i_0) & 1 & i_0 & i_0^2 & \cdots & i_0^{k-3} & x_{i_0} & x_{i_0}^2 \\ p(i_1) & 1 & i_1 & i_1^2 & \cdots & i_1^{k-3} & x_{i_1} & x_{i_1}^2 \\ p(i_2) & 1 & i_2 & i_2^2 & \cdots & i_2^{k-3} & x_{i_2} & x_{i_2}^2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ p(i_k) & 1 & i_k & i_k^2 & \cdots & i_k^{k-3} & x_{i_k} & x_{i_k}^2 \end{vmatrix}_{(k+1)\times(k+1)}.$$

Clearly, $\Delta = 0$ as the first column is the sum of the next $k-2$ columns with the coefficients $\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_{k-3}$.

18. Linearization, Step 2, completed

Expanding Δ by the first column gives the identity

$$\sum_{j=0}^{k}(-1)^j p(i_j)D(I' \setminus \{i_j\}) = 0.$$

Since $p(i_0) = 1$ and $I' \setminus \{i_0\} = I$, the identity rewrites as

$$D(I) = \sum_{j=1}^{k}(-1)^{j+1} p(i_j)D(I' \setminus \{i_j\}),$$

and since $p(i) = 0$ for all $i \in W \setminus \{i_0\}$, all the non-zero summands in the right-hand side are such that $i_j \notin W$.

Expanding Δ by the first column gives the identity

$$\sum_{j=0}^{k}(-1)^j p(i_j) D(I' \setminus \{i_j\}) = 0.$$

$$\Delta = \begin{vmatrix} p(i_0) & 1 & i_0 & i_0^2 & \cdots & i_0^{k-3} & x_{i_0} & x_{i_0}^2 \\ p(i_1) & 1 & i_1 & i_1^2 & \cdots & i_1^{k-3} & x_{i_1} & x_{i_1}^2 \\ p(i_2) & 1 & i_2 & i_2^2 & \cdots & i_2^{k-3} & x_{i_2} & x_{i_2}^2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ p(i_k) & 1 & i_k & i_k^2 & \cdots & i_k^{k-3} & x_{i_k} & x_{i_k}^2 \end{vmatrix}_{(k+1)\times(k+1)}$$

Since $p(i_0) = 1$ and $I' \setminus \{i_0\} = I$, the identity rewrites as

$$D(I) = \sum_{j=1}^{k}(-1)^{j+1} p(i_j) D(I' \setminus \{i_j\}),$$

and since $p(i) = 0$ for all $i \in W \setminus \{i_0\}$, all the non-zero summands in the right-hand side are such that $i_j \notin W$.

Expanding Δ by the first column gives the identity

$$\sum_{j=0}^{k}(-1)^j p(i_j)D(I' \setminus \{i_j\}) = 0.$$

$$\begin{vmatrix} p(i_0) & 1 & i_0 & i_0^2 & \cdots & i_0^{k-3} & x_{i_0} & x_{i_0}^2 \\ p(i_1) & 1 & i_1 & i_1^2 & \cdots & i_1^{k-3} & x_{i_1} & x_{i_1}^2 \\ p(i_2) & 1 & i_2 & i_2^2 & \cdots & i_2^{k-3} & x_{i_2} & x_{i_2}^2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ p(i_k) & 1 & i_k & i_k^2 & \cdots & i_k^{k-3} & x_{i_k} & x_{i_k}^2 \end{vmatrix}_{(k+1)\times(k+1)}$$

Since $p(i_0) = 1$ and $I' \setminus \{i_0\} = I$, the identity rewrites as

$$D(I) = \sum_{j=1}^{k}(-1)^{j+1} p(i_j)D(I' \setminus \{i_j\}),$$

and since $p(i) = 0$ for all $i \in W \setminus \{i_0\}$, all the non-zero summands in the right-hand side are such that $i_j \notin W$.

18. Linearization, Step 2, completed

Expanding Δ by the first column gives the identity

$$\sum_{j=0}^{k}(-1)^j p(i_j)D(I' \setminus \{i_j\}) = 0.$$

$$\begin{vmatrix} p(i_0) & 1 & i_0 & i_0^2 & \cdots & i_0^{k-3} & x_{i_0} & x_{i_0}^2 \\ p(i_1) & 1 & i_1 & i_1^2 & \cdots & i_1^{k-3} & x_{i_1} & x_{i_1}^2 \\ p(i_2) & 1 & i_2 & i_2^2 & \cdots & i_2^{k-3} & x_{i_2} & x_{i_2}^2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ p(i_k) & 1 & i_k & i_k^2 & \cdots & i_k^{k-3} & x_{i_k} & x_{i_k}^2 \end{vmatrix}_{(k+1)\times(k+1)}$$

Since $p(i_0) = 1$ and $I' \setminus \{i_0\} = I$, the identity rewrites as

$$D(I) = \sum_{j=1}^{k}(-1)^{j+1} p(i_j)D(I' \setminus \{i_j\}),$$

and since $p(i) = 0$ for all $i \in W \setminus \{i_0\}$, all the non-zero summands in the right-hand side are such that $i_j \notin W$.

18. Linearization, Step 2, completed

Expanding Δ by the first column gives the identity

$$\sum_{j=0}^{k}(-1)^j p(i_j)D(I' \setminus \{i_j\}) = 0.$$

$$\begin{vmatrix} p(i_0) & 1 & i_0 & i_0^2 & \cdots & i_0^{k-3} & x_{i_0} & x_{i_0}^2 \\ p(i_1) & 1 & i_1 & i_1^2 & \cdots & i_1^{k-3} & x_{i_1} & x_{i_1}^2 \\ p(i_2) & 1 & i_2 & i_2^2 & \cdots & i_2^{k-3} & x_{i_2} & x_{i_2}^2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ p(i_k) & 1 & i_k & i_k^2 & \cdots & i_k^{k-3} & x_{i_k} & x_{i_k}^2 \end{vmatrix}_{(k+1)\times(k+1)}$$

Since $p(i_0) = 1$ and $I' \setminus \{i_0\} = I$, the identity rewrites as

$$D(I) = \sum_{j=1}^{k}(-1)^{j+1} p(i_j)D(I' \setminus \{i_j\}),$$

and since $p(i) = 0$ for all $i \in W \setminus \{i_0\}$, all the non-zero summands in the right-hand side are such that $i_j \notin W$.

18. Linearization, Step 2, completed

Expanding Δ by the first column gives the identity

$$\sum_{j=0}^{k}(-1)^j p(i_j) D(I' \setminus \{i_j\}) = 0.$$

Since $p(i_0) = 1$ and $I' \setminus \{i_0\} = I$, the identity rewrites as

$$D(I) = \sum_{j=1}^{k}(-1)^{j+1} p(i_j) D(I' \setminus \{i_j\}),$$

and since $p(i) = 0$ for all $i \in W \setminus \{i_0\}$, all the non-zero summands in the right-hand side are such that $i_j \notin W$. For each such i_j, we have

$$(I' \setminus \{i_j\}) \setminus W = I' \setminus (W \cup \{i_j\}) = (I \cup \{i_0\}) \setminus (W \cup \{i_j\}) = (I \setminus W) \setminus \{i_j\},$$

whence $|(I' \setminus \{i_j\}) \setminus W| = |I \setminus W| - 1$ and by the inductive assumption, the polynomials $D(I' \setminus \{i_j\})$ are linear combinations of $D(T_1), \ldots, D(T_s)$.

19. Results

Thus, in the step when k states are still to be compressed, the compression can always be achieved by applying a suitable word of length $\leq \binom{n-k+2}{2}$.

Summing up over $k = n, \ldots, 2$, we see that the greedy algorithm always returns a reset word of length $\leq \frac{n^3-n}{6}$:

$$\binom{2}{2} + \binom{3}{2} + \binom{4}{2} + \cdots + \binom{n-1}{2} + \binom{n}{2} =$$

$$\binom{3}{3} + \binom{3}{2} + \binom{4}{2} + \cdots + \binom{n-1}{2} + \binom{n}{2} =$$

$$\binom{4}{3} + \binom{4}{2} + \cdots + \binom{n-1}{2} + \binom{n}{2} = \cdots = \binom{n+1}{3} = \frac{n^3-n}{6}.$$

19. Results

Thus, in the step when k states are still to be compressed, the compression can always be achieved by applying a suitable word of length $\leq \binom{n-k+2}{2}$.
Summing up over $k = n, \dots, 2$, we see that the greedy algorithm always returns a reset word of length $\leq \frac{n^3-n}{6}$:

$$\binom{2}{2} + \binom{3}{2} + \binom{4}{2} + \cdots + \binom{n-1}{2} + \binom{n}{2} =$$

$$\binom{3}{3} + \binom{3}{2} + \binom{4}{2} + \cdots + \binom{n-1}{2} + \binom{n}{2} =$$

$$\binom{4}{3} + \binom{4}{2} + \cdots + \binom{n-1}{2} + \binom{n}{2} = \cdots = \binom{n+1}{3} = \frac{n^3-n}{6}.$$

19. Results

Thus, in the step when k states are still to be compressed, the compression can always be achieved by applying a suitable word of length $\leq \binom{n-k+2}{2}$.
Summing up over $k = n, \dots, 2$, we see that the greedy algorithm always returns a reset word of length $\leq \frac{n^3-n}{6}$:

$$\binom{2}{2} + \binom{3}{2} + \binom{4}{2} + \cdots + \binom{n-1}{2} + \binom{n}{2} =$$

$$\binom{3}{3} + \binom{3}{2} + \binom{4}{2} + \cdots + \binom{n-1}{2} + \binom{n}{2} =$$

$$\binom{4}{3} + \binom{4}{2} + \cdots + \binom{n-1}{2} + \binom{n}{2} = \cdots = \binom{n+1}{3} = \frac{n^3-n}{6}.$$

19. Results

Thus, in the step when k states are still to be compressed, the compression can always be achieved by applying a suitable word of length $\leq \binom{n-k+2}{2}$.
Summing up over $k = n, \ldots, 2$, we see that the greedy algorithm always returns a reset word of length $\leq \frac{n^3-n}{6}$:

$$\binom{2}{2} + \binom{3}{2} + \binom{4}{2} + \cdots + \binom{n-1}{2} + \binom{n}{2} =$$

$$\binom{3}{3} + \binom{3}{2} + \binom{4}{2} + \cdots + \binom{n-1}{2} + \binom{n}{2} =$$

$$\binom{4}{3} + \binom{4}{2} + \cdots + \binom{n-1}{2} + \binom{n}{2} = \cdots = \binom{n+1}{3} = \frac{n^3-n}{6}.$$

Thus, in the step when k states are still to be compressed, the compression can always be achieved by applying a suitable word of length $\leq \binom{n-k+2}{2}$.
Summing up over $k = n, \dots, 2$, we see that the greedy algorithm always returns a reset word of length $\leq \frac{n^3-n}{6}$:

$$\binom{2}{2} + \binom{3}{2} + \binom{4}{2} + \cdots + \binom{n-1}{2} + \binom{n}{2} =$$
$$\binom{3}{3} + \binom{3}{2} + \binom{4}{2} + \cdots + \binom{n-1}{2} + \binom{n}{2} =$$
$$\binom{4}{3} + \binom{4}{2} + \cdots + \binom{n-1}{2} + \binom{n}{2} = \cdots = \binom{n+1}{3} = \frac{n^3-n}{6}.$$

19. Results

Thus, in the step when k states are still to be compressed, the compression can always be achieved by applying a suitable word of length $\leq \binom{n-k+2}{2}$.
Summing up over $k = n, \ldots, 2$, we see that the greedy algorithm always returns a reset word of length $\leq \frac{n^3-n}{6}$:

$$\binom{2}{2} + \binom{3}{2} + \binom{4}{2} + \cdots + \binom{n-1}{2} + \binom{n}{2} =$$

$$\binom{3}{3} + \binom{3}{2} + \binom{4}{2} + \cdots + \binom{n-1}{2} + \binom{n}{2} =$$

$$\binom{4}{3} + \binom{4}{2} + \cdots + \binom{n-1}{2} + \binom{n}{2} = \cdots = \binom{n+1}{3} = \frac{n^3-n}{6}.$$

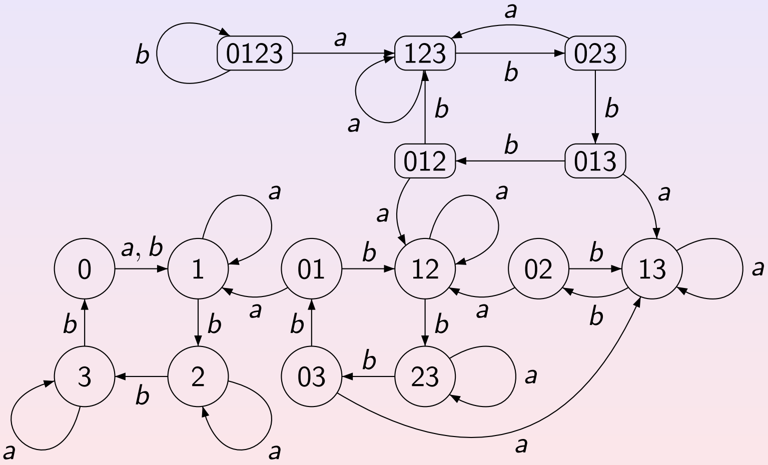
20. Greedy Algorithm

GreedyCompression($\mathscr{A}$)

```
w ← ε                                              ▷ Initializing the current word
P ← Q                                              ▷ Initializing the current set
while |P| > 1 do
    if |P . u| = |P| for all u ∈ Σ* then
        return  Failure
    else
        take a word v ∈ Σ* of minimum length with |P . v| < |P|
    w ← wv                                         ▷ Updating the current word
    P ← P . v                                      ▷ Updating the current set
return  w
```

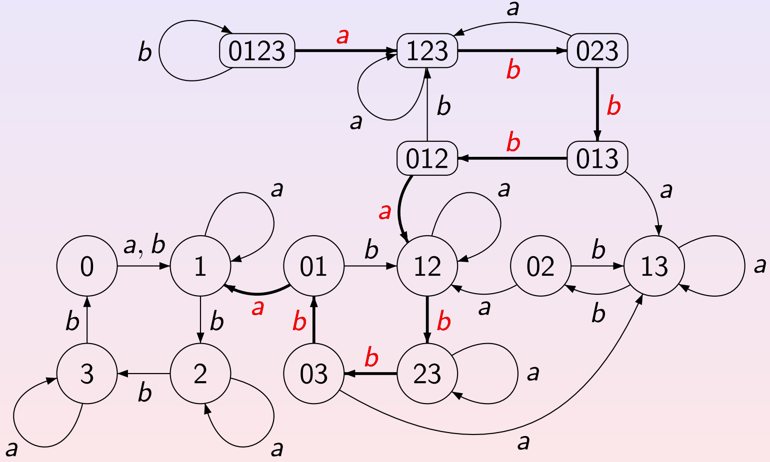
21. Example Revisited

We have already seen that the greedy algorithm fails to find a reset word of minimum length.



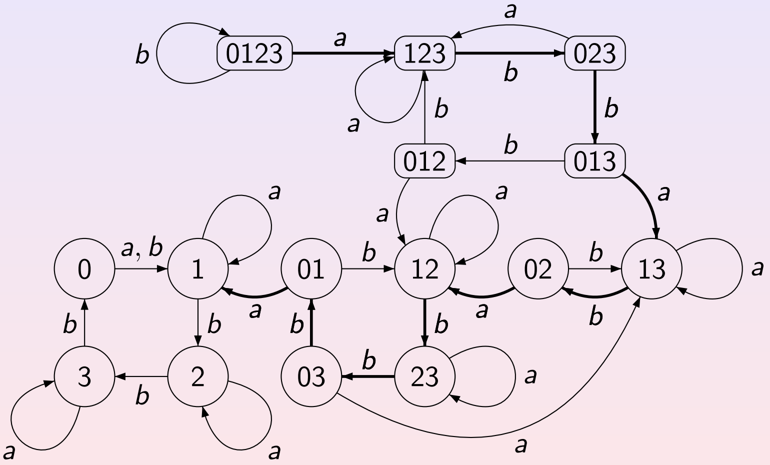
21. Example Revisited

We have already seen that the greedy algorithm fails to find a reset word of minimum length.



21. Example Revisited

We have already seen that the greedy algorithm fails to find a reset word of minimum length.



Actually, the gap between the minimum length of a reset word and the length of the word produced by the greedy algorithm may be arbitrarily large: for each $n > 1$ there exists a synchronizing automaton with n states whose shortest reset word has length $(n-1)^2$ while the greedy algorithm produces a reset word of length $\Omega(n^2 \log n)$.

Very recently, Dmitry Ananichev and Vladimir Gusev (in print) have provided a deep analysis of the worst case behaviour of all natural variants of the greedy algorithm.

The behaviour of the greedy algorithm on average is not yet understood; practically it behaves rather well.

Actually, the gap between the minimum length of a reset word and the length of the word produced by the greedy algorithm may be arbitrarily large: for each $n > 1$ there exists a synchronizing automaton with n states whose shortest reset word has length $(n-1)^2$ while the greedy algorithm produces a reset word of length $\Omega(n^2 \log n)$.

Very recently, Dmitry Ananichev and Vladimir Gusev (in print) have provided a deep analysis of the worst case behaviour of all natural variants of the greedy algorithm.

The behaviour of the greedy algorithm on average is not yet understood; practically it behaves rather well.

22. Greedy Algorithm: Conclusion

Actually, the gap between the minimum length of a reset word and the length of the word produced by the greedy algorithm may be arbitrarily large: for each $n > 1$ there exists a synchronizing automaton with n states whose shortest reset word has length $(n-1)^2$ while the greedy algorithm produces a reset word of length $\Omega(n^2 \log n)$.

Very recently, Dmitry Ananichev and Vladimir Gusev (in print) have provided a deep analysis of the worst case behaviour of all natural variants of the greedy algorithm.

The behaviour of the greedy algorithm on average is not yet understood; practically it behaves rather well.

22. Greedy Algorithm: Conclusion

Actually, the gap between the minimum length of a reset word and the length of the word produced by the greedy algorithm may be arbitrarily large: for each $n > 1$ there exists a synchronizing automaton with n states whose shortest reset word has length $(n-1)^2$ while the greedy algorithm produces a reset word of length $\Omega(n^2 \log n)$.

Very recently, Dmitry Ananichev and Vladimir Gusev (in print) have provided a deep analysis of the worst case behaviour of all natural variants of the greedy algorithm.

The behaviour of the greedy algorithm on average is not yet understood; practically it behaves rather well.