Implementation of the algorithm for testing an automaton for synchronization in linear expected time

Pavel Ageev,
Ural Federal University

Basic notation

A deterministic finite automaton (DFA)

$$\mathcal{A} = (Q, \Sigma, \delta)$$

- $\boldsymbol{Q}$ is a finite set of ***states***
- $\boldsymbol{\Sigma}$ stands for a finite ***alphabet***
- $\boldsymbol{\delta}$ is a ***transition function*** $\delta : Q \times \Sigma \to Q$

Basic notation

A deterministic finite automaton (DFA)

$$\mathcal{A} = (Q, \Sigma, \delta)$$

- $\boldsymbol{Q}$ is a finite set of ***states***
- $\boldsymbol{\Sigma}$ stands for a finite ***alphabet***
- $\boldsymbol{\delta}$ is a ***transition function*** $\delta : Q \times \Sigma \to Q$

$\boldsymbol{\Sigma}^*$ is the set of all words over $\boldsymbol{\Sigma}$

Basic notation

A DFA (Q, Σ, δ) is called ***synchronizing***
if there exist a word $w \in \Sigma^*$ and a state $f \in Q$
such that
$$\delta(q, w) = f \text{ for all } q \in Q$$

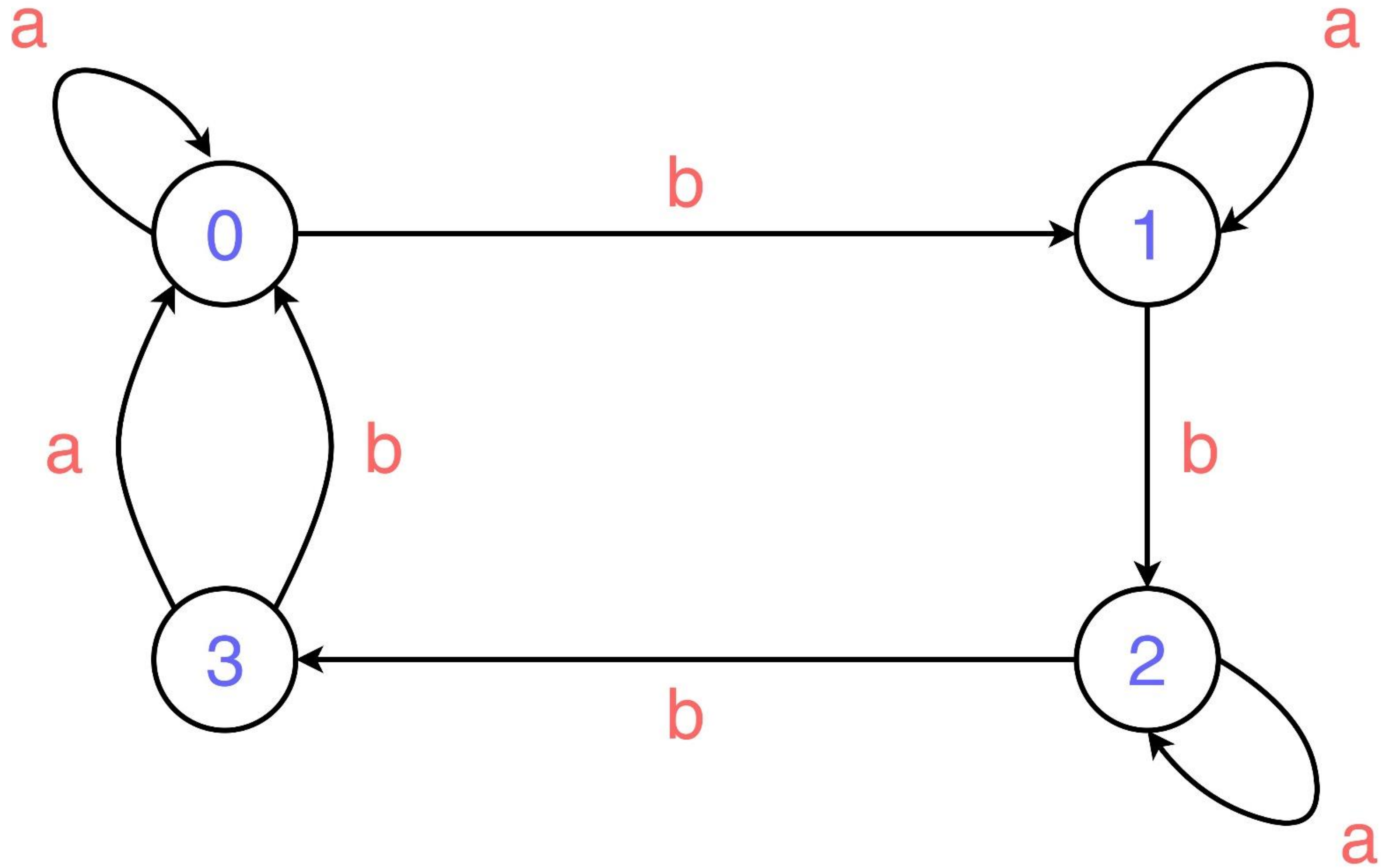
Basic notation

A DFA (Q, Σ, δ) is called ***synchronizing***
if there exist a word $w \in \Sigma^*$ and a state $f \in Q$
such that

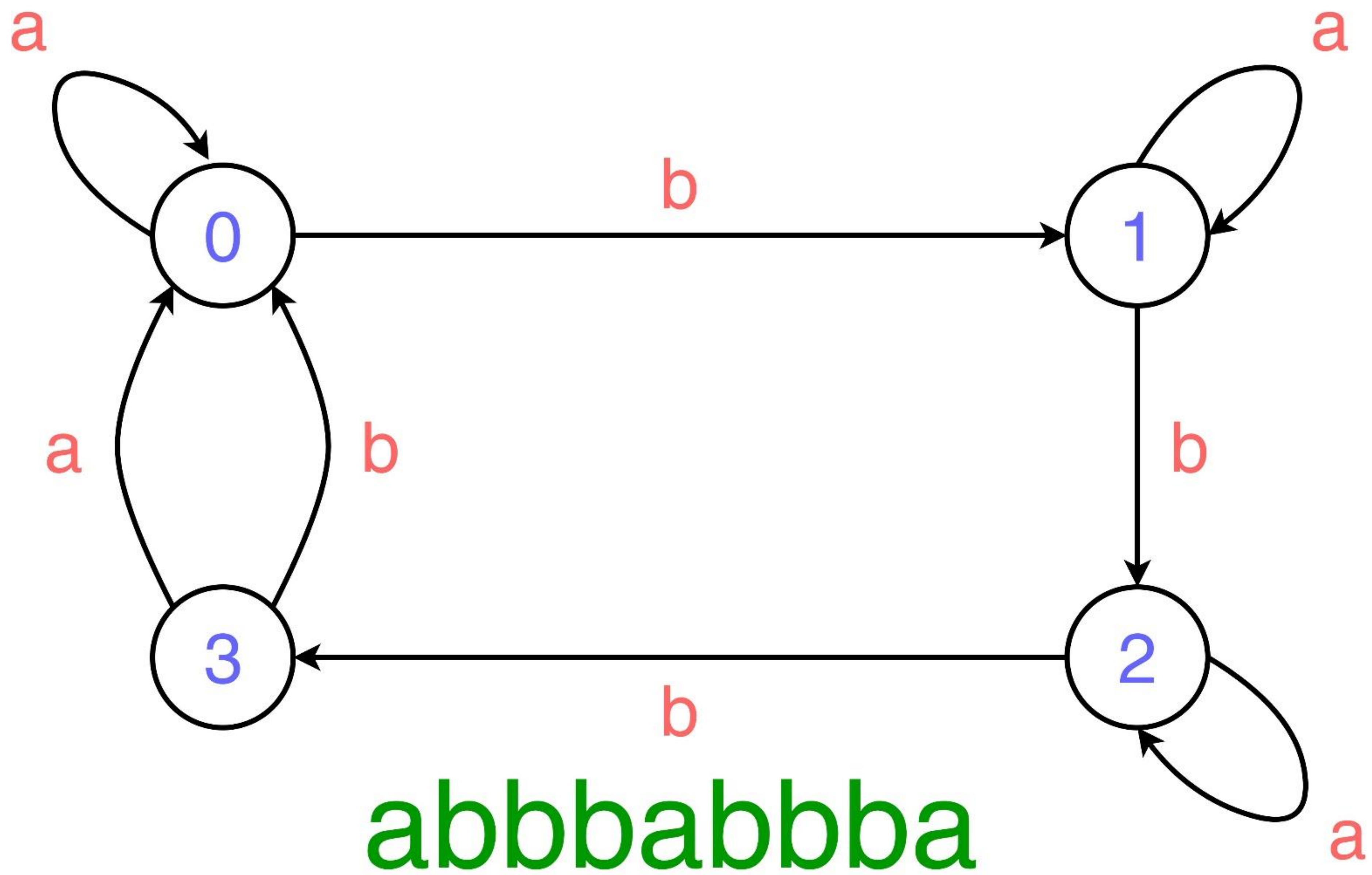
$$\delta(q, w) \ = f \text{ for all } q \in Q$$

Any such w is a ***synchronizing word***

An example



An example



Testing an automaton for synchronization

Černý's synchronization criterion (1964)

An automaton is synchronizing if and only if for each pair of states there exists a word mapping them to the single state.

The implied algorithm requires $\mathbf{\Theta}\left(|\boldsymbol{\Sigma}||\mathbf{Q}|^{2}\right)$ time and memory.

Testing an automaton for synchronization

Černý's synchronization criterion (1964)

An automaton is synchronizing if and only if for each pair of states there exists a word mapping them to the single state.

The implied algorithm requires $\mathbf{O}(|\boldsymbol{\Sigma}||\mathbf{Q}|^2)$ time and memory.

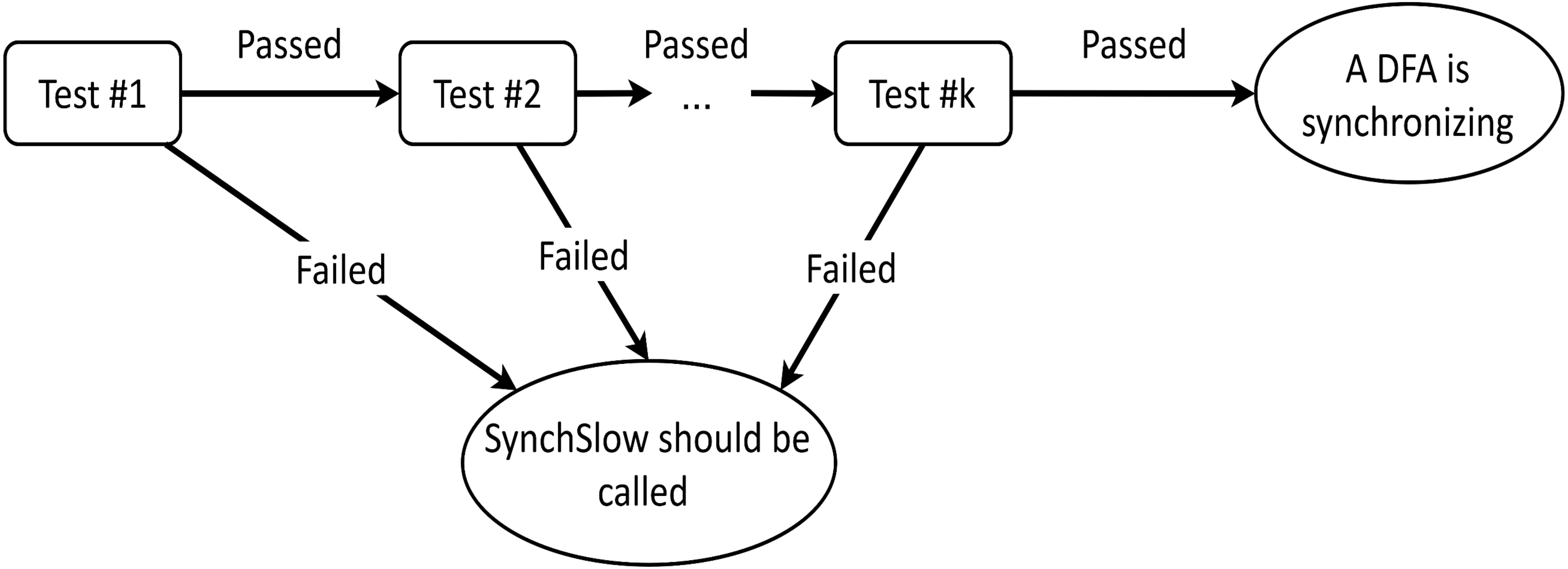
SynchSlow

Berlinkov's algorithm. Theory.

The probability of being synchronizable
for 2-letter random automata with n states is

$$1 - \Theta\left(\frac{1}{n}\right)$$

Berlinkov's algorithm. Theory.



Berlinkov’s algorithm. Theory.

Each test has two features:

Berlinkov’s algorithm. Theory.

Each condition has two features:

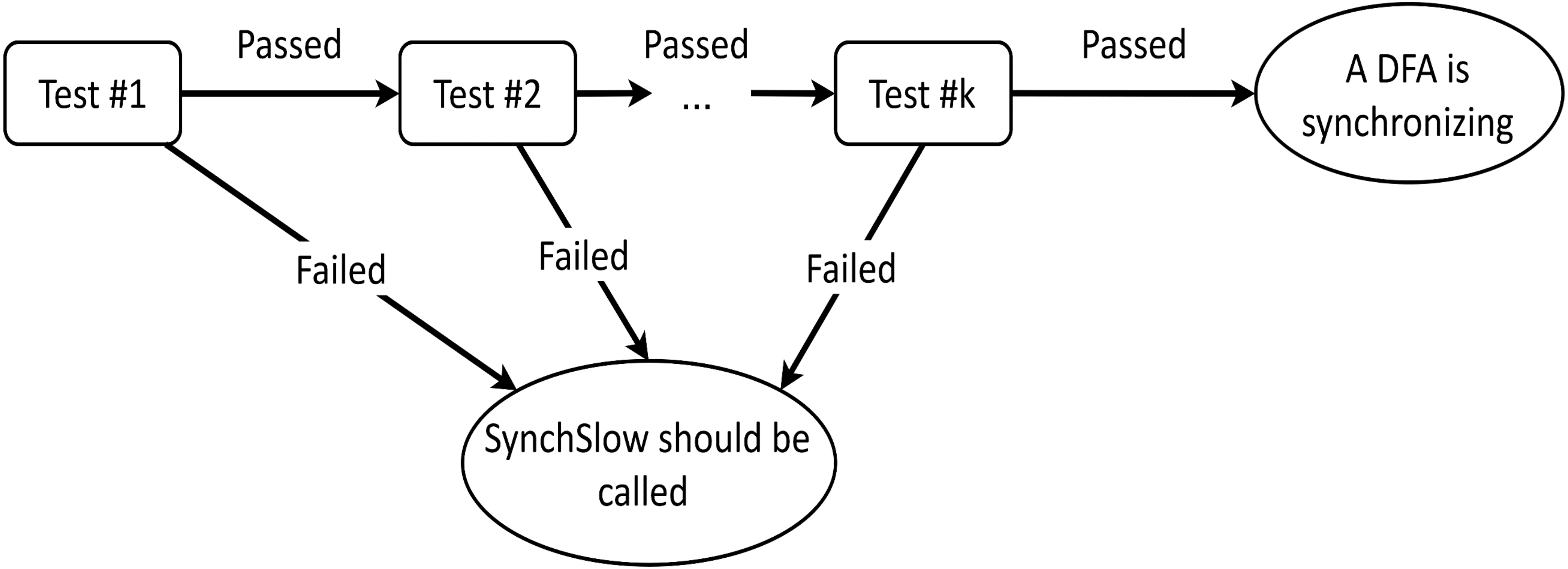
1. it can be checked in $O(n)$ in every DFA with n states;

Berlinkov's algorithm. Theory.

Each condition has two features:

1. it can be checked in $O(n)$ in every DFA with n states;
2. the fraction of automata with n states not satisfying the condition is $O\left(\frac{1}{n}\right)$.

Berlinkov's algorithm. Theory.



Berlinkov's algorithm. Theory.

The expected time of the whole algorithm will be

$$O(n)\left(1 - O\left(\frac{1}{n}\right)\right) + O(n^2)O\left(\frac{1}{n}\right) = O(n).$$

Berlinkov’s algorithm. Practice.

If you implement Berlinkov’s algorithm as is,
it will invoke *SyncSlow*
for **EVERY** automaton with n
up to **SEVERAL THOUSAND**!

Berlinkov’s algorithm. Practice.

Why?

Berlinkov's algorithm. Practice.

The probability of being synchronizable for 2-letter random automata with n states equals $1 - \Theta\left(\frac{min\{n, 10000\}}{n}\right)$.

Berlinkov's algorithm. Practice.

$$1 - \Theta\left(\frac{min\{n,10000\}}{n}\right)$$

$$\Downarrow$$

My contribution

$$1 - \Theta\left(\frac{min\{n,6\}}{n}\right)$$

Berlinkov’s algorithm. Practice.

10000 ⇒ 35

Are you interested?

Welcome to the poster section

Feel free to mail me (birne.ageev@gmail.com)

Are you NOT interested?