Quantum Finite Automata and Their Simulations

Gustaw Lippa

Krzysztof Makieła

Marcin Kuta
mkuta@agh.edu.pl

Department of Computer Science,
AGH University of Science and Technology, Kraków, Poland

Kraków Quantum Informatics Seminar (KQIS)
13 October 2020

Outline

- Motivation
- Quantum finite automata
- Library for simulating quantum finite automata

Existing libraries

- **Java Formal Languages and Automata Package** (JFLAP)
- Quirk
- Quantum++
- Q#
- Qiskit
- ProjectQ

Types of Finite Automata

- Classical automata
  - **Deterministic Finite Automaton** (DFA)
  - **Nondeterministic Finite Automaton** (NFA)
  - **Alternating Finite Automaton** (AFA)

Types of Finite Automata

- Classical automata
  - **Deterministic Finite Automaton** (DFA)
  - **Nondeterministic Finite Automaton** (NFA)
  - **Alternating Finite Automaton** (AFA)
- Probabilistic automata
  - **Probabilistic Finite Automaton** (PFA)

Types of Finite Automata

- Classical automata
  - **Deterministic Finite Automaton** (DFA)
  - **Nondeterministic Finite Automaton** (NFA)
  - **Alternating Finite Automaton** (AFA)
- Probabilistic automata
  - **Probabilistic Finite Automaton** (PFA)
- Quantum automata
  - **Measure-Once Quantum Finite Automaton** (MO-QFA)
  - **Measure-Many Quantum Finite Automaton** (MM-QFA)
  - **General Quantum Finite Automaton** (GQFA)

Deterministic Finite Automaton

Deterministic Finite Automaton (DFA)

$A = (\Sigma, Q, q_0, Q_{\text{acc}}, \delta)$

Σ - alphabet, finite set of symbols

Q - finite set of states

q_0 - initial state, $q_0 \in Q$

F - set of accepting states, $F \subseteq Q$

$\delta\colon Q \times \Sigma \to Q$ - transition function

Determinism condition

For all $q \in Q$ we have $\sum_{p \in Q} \delta(q, \sigma, p) = 1$

Deterministic Finite Automaton

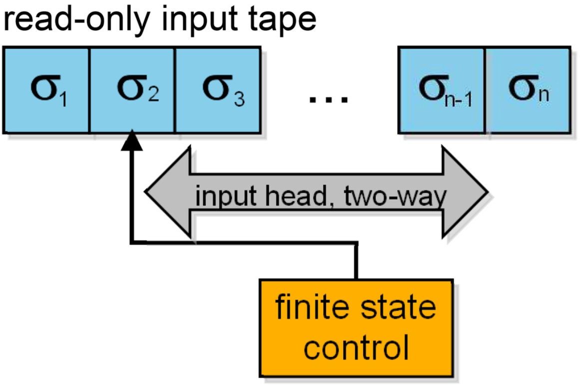


Figure: Internals of Finite Automaton [3]

Deterministic Finite Automaton

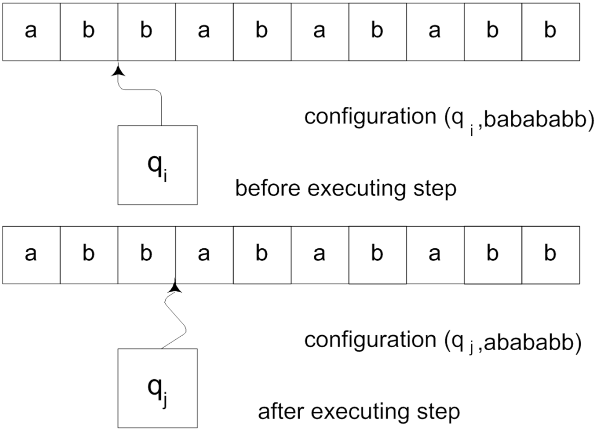


Figure: Configuration of Finite Automaton [2]

Deterministic Finite Automaton

Example

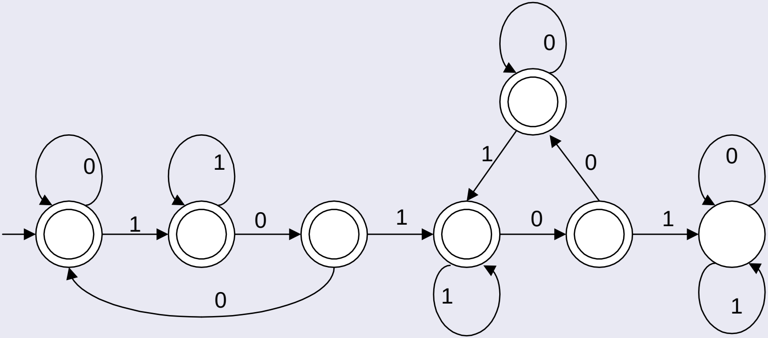


Figure: Deterministic finite automaton [2]

Nondeterministic Finite Automaton

Nondeterministic Finite Automaton (NFA)

$A = (\Sigma, Q, q_0, Q_{\text{acc}}, \delta)$

Σ - alphabet, finite set of symbols

Q - finite set of states

q_0 - initial state, $q_0 \in Q$

F - set of accepting states, $F \subseteq Q$

$\delta\colon Q \times \Sigma \to 2^Q$ - transition function

Nondeterministic Finite Automaton

Example

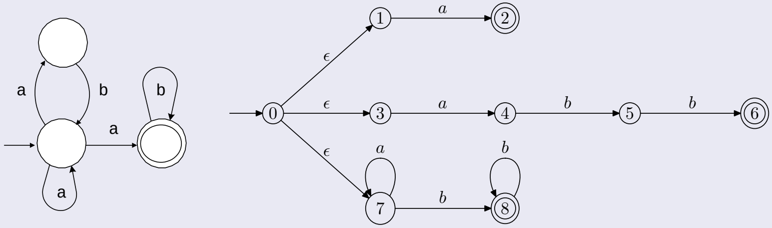


Figure: Nondeterministic finite automaton [2]

Probabilistic Finite Automaton

Probabilistic Finite Automaton (PFA)

$A = (\Sigma, Q, \pi, \eta, \{M_\sigma\}_{\sigma \in \Sigma})$

Σ - alphabet, finite set of symbols

π - vector denoting initial distribution of states, $\pi \in [0,1]^{1 \times N}$

η - vector encoding accepting states, $\eta \in \{0,1\}^{N \times 1}$

$\{M_\sigma\}_{\sigma \in \Sigma}$ - set of transition matrices

Acceptance probability

$$w = \sigma_1 \ldots \sigma_n$$

$$P_A(w) = \pi M(\sigma_1) \ldots M(\sigma_n) \eta$$

Measure-Once Quantum Finite Automaton

Measure-Once Quantum Finite Automaton (MO-QFA)

$A = (Q, \Sigma, q_0, F, \{U_\sigma\}_{\sigma \in \Sigma})$

Σ - alphabet, finite set of symbols

Q - finite set of states

q_0 - initial state, $q_0 \in Q$

F - set of accepting states, $F \subseteq Q$

$\{U_\sigma\}_{\sigma \in \Sigma}$ - set of transition matrices

Acceptance probability

$$w = \sigma_1 \dots \sigma_n$$

$$P_A(w) = \|P_{\text{acc}} U(\sigma_n) \dots U(\sigma_1) |q_0\rangle\|^2$$

(Moore, C., Crutchfield, J.P.: *Quantum automata and quantum grammars.* Theoretical Computer Science **237**(1-2), 275–306 (2000))

Measure-Once Quantum Finite Automaton

Dual formulation of automaton

	Classical	Matrix
Initial state	q_0	$\lvert q_0\rangle = (1, 0, \ldots, 0)^T$
Transitions	$\delta\colon Q \times \Sigma \times Q \to \mathbb{C}$	$\{U_\sigma\}_{\sigma\in\Sigma}$
	$\delta(q_i, \sigma, q_j)$	$U_\sigma(j, i)$
Accepting states	F	$P_{\mathrm{acc}} = \sum_{q\in F} \lvert q\rangle\langle q\rvert$

Unitarity condition

Transition function δ:

$$\sum_{p\in Q} \overline{\delta(q_1, \sigma, p)}\delta(q_2, \sigma, p) = \begin{cases} 1 & q_1 = q_2 \\ 0 & q_1 \neq q_2 \end{cases}$$

Transition matrix U:

$$U_\sigma^\dagger U_\sigma = U_\sigma U_\sigma^\dagger = I_{|Q|}$$

Pumping lemma

Theorem (Pumping lemma for regular languages)

Let language L be a regular language.

Then there exists constant K such that

for all $w \in L$, $|w| \geq K$

there exist x, y, z such that

(1) $w = xyz$

(2) $|xy| \leq K$

(3) $|y| \geq 1$

(4) for all $i \geq 0$ it holds $w_i = xy^i z \in L$

Pumping lemma

Theorem (Pumping lemma for quantum regular languages)

Let language L be recognized by $MO - QFA$.

Then there exists constant K such that

for any w and any $\varepsilon > 0$

for any u, v

it holds $|P_A(xy^K z) - P_A(xyz)| < \varepsilon$.

Additionally, if automaton A is n-dimensional there exists constant c such that

$K < (c\varepsilon)^{-n}$

Measure-Once Quantum Finite Automaton

Example

$A = (Q, \Sigma, q_0, F, \{U_\sigma\}_{\sigma \in \Sigma})$
$\Sigma = \{a\}$
$Q = \{q_0, q_1\}$
$F = \{q_1\}$
$\delta:$
$\delta(q_0, a, q_0) = \frac{1}{\sqrt{2}} \quad \delta(q_0, a, q_1) = \frac{1}{\sqrt{2}}$
$\delta(q_1, a, q_0) = \frac{1}{\sqrt{2}} \quad \delta(q_1, a, q_1) = -\frac{1}{\sqrt{2}}$

Matrix formulation

$$|q_0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad |q_1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$U(a) = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \quad P_{\text{acc}} = |q_1\rangle\langle q_1| = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

Measure-Many Quantum Finite Automaton

Measure-Many Quantum Finite Automaton (MM-QFA)

$A = (Q, \Sigma, q_0, Q_{acc}, Q_{rej}, \{U_\sigma\}_{\sigma\in\Gamma})$

Σ - alphabet, finite set of symbols

Q - finite set of states

q_0 - initial state, $q_0 \in Q$

Q_{acc} - set of accepting states, $Q_{acc} \subseteq Q$

Q_{rej} - set of rejecting states, $Q_{rej} \subseteq Q$

$\{U_\sigma\}_{\sigma\in\Sigma}$ - set of transition matrices

Evolution

$|\Psi\rangle \to P_{non}|U|\Psi\rangle$
$p_{acc} \to p_{acc} + \||P_{acc}|U|\Psi\rangle\|^2$
$p_{rej} \to p_{rej}p_{acc} + \|P_{rej}|U|\Psi\rangle\|^2$

(Kondacs, A., Watrous, J.: *On the power of quantum finite state automata.* 38th Annual Symposium on Foundations of Computer Science, FOCS'97)

Measure-Many Quantum Finite Automaton

Acceptance probability

$$w = \sigma_1 \ldots \sigma_n$$

$$P_A(w) = \sum_{k=1}^{n+1} \| P_{\text{acc}} U(\sigma_k) \prod_{i=1}^{k-1} (P_{\text{non}} U(\sigma_i)) \|^2$$

Example

Example

$A = (Q, \Sigma, q_0, Q_{acc}, Q_{rej}, \{U_\sigma\}_{\sigma \in \Sigma})$
$\Sigma = \{a\}$
$Q = \{q_0, q_1, q_{acc}, q_{rej}\}$
$Q_{acc} = \{q_{acc}\}$
$Q_{rej} = \{q_{rej}\}$
δ:
$\delta(q_0, a, q_0) = \frac{1}{2}$ $\quad \delta(q_0, a, q_1) = \frac{1}{\sqrt{2}}$
$\delta(q_0, a, q_{acc}) = \frac{1}{2}$ $\quad \delta(q_0, a, q_{rej}) = 0$
...

Evolution

$U(a)|q_0\rangle = \frac{1}{2}|q_0\rangle + \frac{1}{\sqrt{2}}|q_1\rangle + \frac{1}{2}|q_{acc}\rangle$
$U(a)|q_1\rangle = \frac{1}{2}|q_0\rangle - \frac{1}{\sqrt{2}}|q_1\rangle + \frac{1}{2}|q_{acc}\rangle$
$U(\$)|q_0\rangle = |q_{acc}\rangle$
$U(\$)|q_1\rangle = |q_{rej}\rangle$

Comparison of MO-QFA and MM-QFA

MO-QFA	MM-QFA
One measurement	Many measurements
after reading the last symbol	after reading each symbol
acceptance or rejection	acceptance, rejection or continuation

Advantages and disadvantages of QFA

- QFA can be exponetially more space efficent than DFA or PFA
- Sometimes it is impossible to simulate DFA by QFA (due to limited memory)
- QFA cannot recognize all regular languages (due to reversibility)

General Quantum Finite Automaton

General Quantum Finite Automaton (GQFA)

$A = (Q, \Sigma, q_0, Q_{acc}, Q_{rej}, \{U_\sigma\}_{\sigma \in \Gamma})$

Σ - alphabet, finite set of symbols

Q - finite set of states

q_0 - initial state, $q_0 \in Q$

Q_{acc} - set of accepting states, $Q_{acc} \subseteq Q$

Q_{rej} - set of rejecting states, $Q_{rej} \subseteq Q$

$\{U_\sigma\}_{\sigma \in \Sigma}$ - set of transition matrices

Two Formulations of Finite Automata

Automaton	Transition function	Transition matrix

Two Formulations of Finite Automata

Automaton	Transition function	Transition matrix
NFA	$\delta\colon Q \times \Sigma \times Q \to \{0,1\}$	$M_\sigma \in \{0,1\}^{\lvert Q \rvert \times \lvert Q \rvert}$

Two Formulations of Finite Automata

Automaton	Transition function	Transition matrix
NFA	$\delta\colon Q \times \Sigma \times Q \to \{0,1\}$	$M_\sigma \in \{0,1\}^{\lvert Q\rvert \times \lvert Q\rvert}$
DFA	$\delta\colon Q \times \Sigma \times Q \to \{0,1\}$	$M_\sigma \in \{0,1\}^{\lvert Q\rvert \times \lvert Q\rvert}$
	$\sum_{p \in Q} \delta(q, \sigma, p) = 1$	$M_\sigma \mathbf{1} = \mathbf{1}$

Two Formulations of Finite Automata

Automaton	Transition function	Transition matrix
NFA	$\delta\colon Q \times \Sigma \times Q \to \{0,1\}$	$M_\sigma \in \{0,1\}^{\lvert Q\rvert \times \lvert Q\rvert}$
DFA	$\delta\colon Q \times \Sigma \times Q \to \{0,1\}$	$M_\sigma \in \{0,1\}^{\lvert Q\rvert \times \lvert Q\rvert}$
	$\sum_{p \in Q} \delta(q,\sigma,p) = 1$	$M_\sigma \mathbf{1} = \mathbf{1}$
PFA	$\delta\colon Q \times \Sigma \times Q \to [0,1]$	$M_\sigma \in [0,1]^{\lvert Q\rvert \times \lvert Q\rvert}$
	$\sum_{p \in Q} \delta(q,\sigma,p) = 1$	$M_\sigma \mathbf{1} = \mathbf{1}$

Two Formulations of Finite Automata

Automaton	Transition function	Transition matrix
NFA	$\delta\colon Q \times \Sigma \times Q \to \{0,1\}$	$M_\sigma \in \{0,1\}^{\lvert Q\rvert \times \lvert Q\rvert}$
DFA	$\delta\colon Q \times \Sigma \times Q \to \{0,1\}$ $\sum_{p \in Q} \delta(q,\sigma,p) = 1$	$M_\sigma \in \{0,1\}^{\lvert Q\rvert \times \lvert Q\rvert}$ $M_\sigma \mathbf{1} = \mathbf{1}$
PFA	$\delta\colon Q \times \Sigma \times Q \to [0,1]$ $\sum_{p \in Q} \delta(q,\sigma,p) = 1$	$M_\sigma \in [0,1]^{\lvert Q\rvert \times \lvert Q\rvert}$ $M_\sigma \mathbf{1} = \mathbf{1}$
MO-QFA	$\delta\colon Q \times \Sigma \times Q \to \mathbb{C}$ $\sum_{p \in Q} \overline{\delta(q_1,\sigma,p)}\delta(q_2,\sigma,p) = \delta_{q_1=q_2}$	$M_\sigma \in \mathbb{C}^{\lvert Q\rvert \times \lvert Q\rvert}$ $U_\sigma^\dagger U_\sigma = U_\sigma U_\sigma^\dagger = I_{\lvert Q\rvert}$

Two Formulations of Finite Automata

Automaton	Transition function	Transition matrix
NFA	$\delta\colon Q \times \Sigma \times Q \to \{0,1\}$	$M_\sigma \in \{0,1\}^{\lvert Q\rvert \times \lvert Q\rvert}$
DFA	$\delta\colon Q \times \Sigma \times Q \to \{0,1\}$ $\sum_{p \in Q} \delta(q,\sigma,p) = 1$	$M_\sigma \in \{0,1\}^{\lvert Q\rvert \times \lvert Q\rvert}$ $M_\sigma \mathbf{1} = \mathbf{1}$
PFA	$\delta\colon Q \times \Sigma \times Q \to [0,1]$ $\sum_{p \in Q} \delta(q,\sigma,p) = 1$	$M_\sigma \in [0,1]^{\lvert Q\rvert \times \lvert Q\rvert}$ $M_\sigma \mathbf{1} = \mathbf{1}$
MO-QFA	$\delta\colon Q \times \Sigma \times Q \to \mathbb{C}$ $\sum_{p \in Q} \overline{\delta(q_1,\sigma,p)}\delta(q_2,\sigma,p) = \delta_{q_1=q_2}$	$M_\sigma \in \mathbb{C}^{\lvert Q\rvert \times \lvert Q\rvert}$ $U_\sigma^\dagger U_\sigma = U_\sigma U_\sigma^\dagger = I_{\lvert Q\rvert}$
MM-QFA	$\delta\colon Q \times \Sigma \times Q \to \mathbb{C}$ $\sum_{p \in Q} \overline{\delta(q_1,\sigma,p)}\delta(q_2,\sigma,p) = \delta_{q_1=q_2}$	$M_\sigma \in \mathbb{C}^{\lvert Q\rvert \times \lvert Q\rvert}$ $U_\sigma^\dagger U_\sigma = U_\sigma U_\sigma^\dagger = I_{\lvert Q\rvert}$

Language Acceptance Modes 1/2

- *with a cut-point* $\lambda \in [0, 1)$, if for all $x \in L$, we have $P_A(x) > \lambda$ and for all $x \notin L$, we have $P_A(x) \leq \lambda$. This mode of acceptance is also called *with an unbounded error*.
- *with an isolated cut-point* $\lambda \in [0, 1)$, if there exists $\varepsilon \geq 0$, such, that for all $x \in L$, we have $P_A(x) \geq \lambda + \varepsilon$ and for all $x \notin L$, we have $P_A(x) \leq \lambda - \varepsilon$.
- *with a bounded error* $\varepsilon \in [0, \frac{1}{2})$, if for all $x \in L$, we have $P_A(x) \geq 1 - \varepsilon$ and for all $x \notin L$, we have $P_A(x) \leq \varepsilon$. This mode of acceptance is equivalent to acceptance with an isolated cut-point, where cut-point $\lambda = \frac{1}{2}$ is isolated with value $\frac{1}{2} - \varepsilon$.

Language Acceptance Modes 2/2

- *with a positive one-sided unbounded error* if for all $x \in L$, we have $P_A(x) > 0$.
- *with a negative one-sided unbounded error* if for all $x \in L$, we have $P_A(x) = 1$.
- *Monte Carlo acceptance*, if there exists $\varepsilon \in (0, \frac{1}{2}]$ such, that for all $x \in L$, we have $P_A(x) = 1$ and for all $x \notin L$, we have $P_A(x) \leq \varepsilon$. Such A is called Monte Carlo *QFA* for L.

Hierarchy of quantum languages

Hierarchy of quantum languages

Automaton	Class of languages	
DFA	Regular	
NFA	Regular	
	Acceptance	
	Bounded	Unbounded
PFA	Regular	Stochastic
MO-QFA	$\subset$ Regular	$\subset$ Stochastic
MM-QFA	$\subset$ Regular	Stochastic
GQFA	$\subset$ Regular	
QFAC	Regular	
CL-QFA	Regular	

Hierarchy of quantum automata

Hierarchy of quantum automata

	Once	Many
$U(\sigma)$	MO-QFA	MM-QFA
$U(\sigma)$+Projective Measurement	LQFA	GQFA

Hierarchy of quantum languages

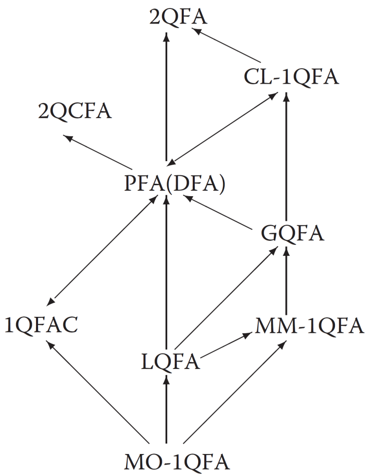


Figure: Hierarchy of classes of quantum languages recognized with bounded error [7]

Functionality of the library

- Definitions of automata: PFA, MO-QFA, MM-QFA, GQFA
- Simulations of automaton for a given word
- Sample generation from a language defined by regular expressions
- Examination of different acceptance modes for a given language
- Results visualisation

Problems

Implementation problems

- Rounding errors
- Stability
- Samples generation
- Computational complexity

Possible solutions

- Rounding errors
  - More precise representation of complex numbers
  - Estimating generated error based on performed computation
  - A custom solution developed using domain knowledge
- Stability
  - Results verification
- Samples generation
  - Using an existing library
  - Generating random samples
- Computational complexity
  - Changing the simulation type from strong to weak
  - Optimising computation
  - Using existing optimised solutions

Solutions taken in the library

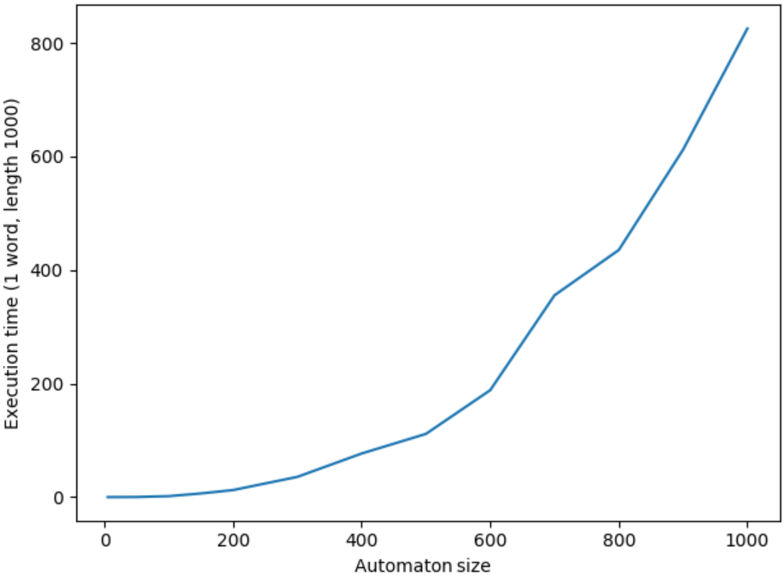
- Rounding errors - errors are estimated although not as precisely as it is probably possible. Future solution may check known constraints (like unitary state matrices) at every step and fix errors during the simulation
- Stability - there is a check at the end of the simulation whether the result is sensible
- Samples generation - samples are generated randomly, skewed towards shorter words. The parameters are configurable
- Computational complexity - the library uses NumPy, which is optimised enough for matrix multiplication, as it's basis

Simulations

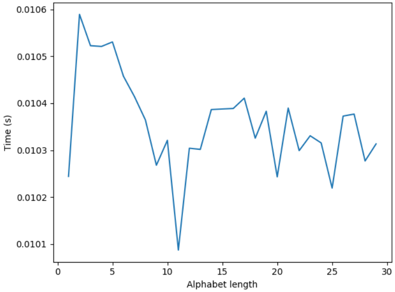
Computational complexity

- Automaton size
- Alphabet size
- Word length

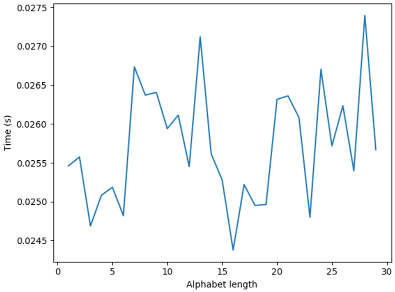
Computational complexity



Alphbet size

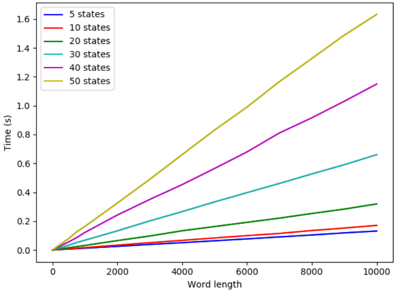


20 states

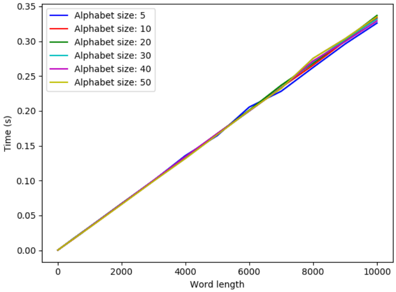


30 states

Word length

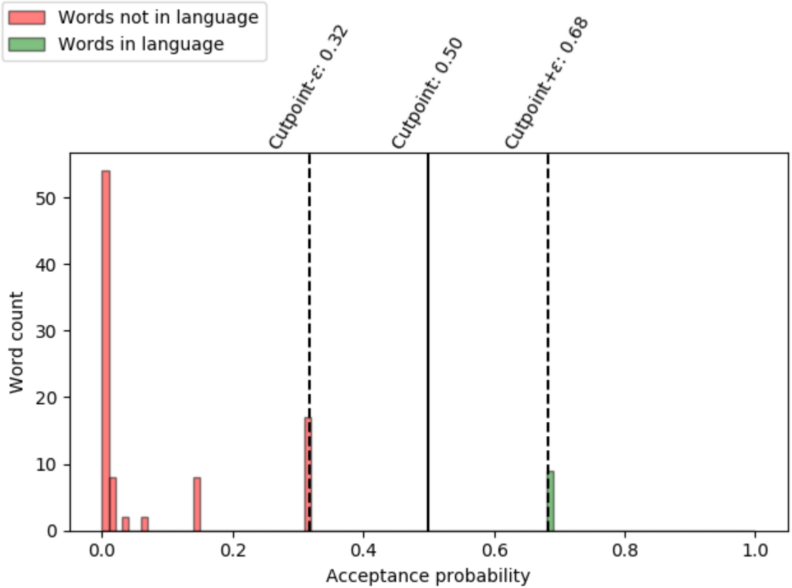


5-letter alphabet



20 states of automaton

Results visualisation



Usage example 1/4

```python
import numpy as np
from math import sqrt
from QFA.MM_1QFA import MM_1QFA
from QFA.LanguageGenerator import LanguageGenerator
from QFA.LanguageChecker import LanguageChecker
from QFA.Plotter import Plotter
```

Usage example 2/4

```
alphabet = 'ab'

p = 0.682327803828019  # Auxillary variable

# Initial state of automaton
initial_state = np.array([[sqrt(1-p)], [sqrt(p)], [0], [0]])

# Transition matrices
U_a = np.array([[1-p,            sqrt(p*(1-p)), 0, -sqrt(p) ],
                [sqrt(p*(1-p)), p,              0, sqrt(1-p)],
                [0,              0,              1, 0        ],
                [sqrt(p),       -sqrt(1-p),     0, 0        ]])

U_b = np.array([[0, 0, 0, 1],
                [0, 1, 0, 0],
                [0, 0, 1, 0],
                [1, 0, 0, 0]])

U_end = np.array([[0, 0, 0, 1],
                  [0, 0, 1, 0],
                  [0, 1, 0, 0],
                  [1, 0, 0, 0]])
```

Usage example 3/4

```
# Accepting and rejecting states are defined with matrices
# representing projective measurements
P_acc = np.array([[0, 0, 0, 0],
                  [0, 0, 0, 0],
                  [0, 0, 1, 0],
                  [0, 0, 0, 0]])

P_rej = np.array([[0, 0, 0, 0],
                  [0, 0, 0, 0],
                  [0, 0, 0, 0],
                  [0, 0, 0, 1]])

qfa = MM_1QFA(alphabet, initial_state,
              [U_a, U_b, U_end], P_acc, P_rej)
```

Usage example 4/4

```
language_generator = LanguageGenerator('a*b*',
    alphabet)
language, not_in_language = language_generator.
    get_language_sample()

language_checker = LanguageChecker(qfa, language,
    not_in_language)

plotter = Plotter(language_checker)
plotter.plot()
```

Two-way Quantum Finite Automaton

Two-way Quantum Finite Automaton (2QFA)

$A = (Q, \Sigma, q_0, Q_{\text{acc}}, Q_{\text{rej}}, \delta)$

Σ - alphabet, finite set of symbols

Q - finite set of states

q_0 - initial state, $q_0 \in Q$

Q_{acc} - set of accepting states, $Q_{\text{acc}} \subseteq Q$

Q_{rej} - set of rejecting states, $Q_{\text{rej}} \subseteq Q$

δ - transition function

Two-way Quantum Finite Automaton

Conditions on δ

Local probability and orthogonality condition:

$$\sum_{p\in Q,d} \overline{\delta(q_1,\sigma,p,d)}\delta(q_2,\sigma,p,d) = \begin{cases} 1 & q_1 = q_2 \\ 0 & q_1 \neq q_2 \end{cases}$$

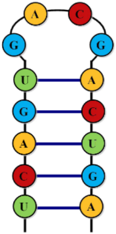
Separability condition I:

$$\sum_{p\in Q} \Big(\overline{\delta(q_1,\sigma,p,\rightarrow)}\delta(q_2,\sigma,p,\downarrow) + \\ + \overline{\delta(q_1,\sigma,p,\downarrow)}\delta(q_2,\sigma,p,\leftarrow) \Big) = 0$$

Separability condition II:

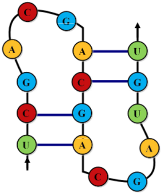
$$\sum_{p\in Q} \overline{\delta(q_1,\sigma,p,\rightarrow)}\delta(q_2,\sigma,p,\leftarrow) = 0$$

Applications of 2QFA



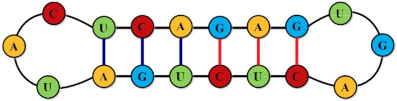
$L = \{ x \in \{A, C, G, U\}^* \mid x = x^R \}$

(a) hairpin loop



$L = \{ A^n G^m U^n C^n \mid n, m \geq 0 \}$

(b) pseudoknot



$L = \{ A^n U^n G^m C^m \mid n, m \geq 0 \}$

(c) dumbbell

Figure: RNA structures recognised by 2QFA [3]

Bibliography I

[1] Quantum finite automata library.
`https://github.com/gustawlippa/QFA`.

[2] Theory of finite automata and formal languages (in polish).
`https://www.kompilatory.agh.edu.pl`.

[3] Amandeep Singh Bhatia and Shenggen Zheng.
RNA-2QCFA: evolving two-way quantum finite automata with classical states for RNA secondary structures.
CoRR, abs/2007.06273, 2020.

[4] Attila Kondacs and John Watrous.
On the power of quantum finite state automata.
In *38th Annual Symposium on Foundations of Computer Science, FOCS'97*, pages 66–75, 1997.

Bibliography II

[5] Gustaw Lippa, Krzysztof Makieła, and Marcin Kuta.
Simulations of quantum finite automata.
In *Proceedings of 20th International Conference on Computational Science, ICCS 2020*, volume 12142 of *LNCS*, pages 441–450, 2020.

[6] Cristopher Moore and James P. Crutchfield.
Quantum automata and quantum grammars.
Theoretical Computer Science, 237(1-2):275–306, 2000.

[7] Daowen Qiu, Lvzhou Li, Paulo Mateus, and Jozef Gruska.
Quantum finite automata.
In *Handbook of Finite State Based Models and Applications*, pages 113–144. 2012.