Recognizability versus solvability of promise problems in finite, classical and quantum automata, framework

Jozef Gruska\textsuperscript{1}, Lvzhou Li\textsuperscript{2}, Shenggen Zheng\textsuperscript{1,*}

\textsuperscript{1}Faculty of Informatics, Masaryk University, Brno 60200, Czech Republic
\textsuperscript{2}Department of Computer Science, Sun Yat-sen University, Guangzhou 510006, China

Abstract

In pioneering papers \cite{13,17}, the concept of the promise problems was introduced and started to be systematically explored. It has been argued that promise problems should be seen as partial decision problems and as such that they are more fundamental than decision problems and formal languages that used to be considered as the basic ones for complexity theory issues explorations. Moreover, both of the above papers explored and summarized \cite{17} in some depth and systematically, promise problems in the context of the theory of the main computational complexity classes as well as in the context of cracking of the public key cryptography.

In the present paper a variety of issues is explored concerning dealing with the promise problems on the level of finite, classical, quantum and also semi-quantum automata. Two acceptance modes, recognizability and solvability are introduced and their basic properties are explored. This is also to capture and explore the case that even in the case the promise is simple (say regular), its disjoint subsets for so called yes and no cases do not have to be so. In addition, some results concerning descriptional complexity impacts on outcomes of some operations on promise problems are shown as well as the significant power of quantum versions of classical automata is demonstrated when dealing with some promise problems.

Keywords: Promise problem, Finite automata, Quantum finite automata, Recognizability, Solvability

1. Introduction

Informally, a promise problem is the problem to decide whether an object or process has a property $P_1$ or $P_2$, provided it is promised (known) to have a property $P_3$.

The concept of a promise problem was introduced explicitly in \cite{13} and it has been argued there that promise problems are actually more fundamental for the study of computational theory issues than decision problems or, more formally, formal language versions/encodings of the decision problems.

Such a view on the fundamental importance of promise problems have been even more emphasized in the survey paper \cite{17}, where also the following basic version of the promise problems has been introduced.

Definition 1. A promise problem over an alphabet $\Sigma$ is a pair $(A_{\text{yes}}, A_{\text{no}})$ of disjoint subsets of $\Sigma^*$. The union $A_{\text{yes}} \cup A_{\text{no}}$ is then called the promise and $A_{\text{yes}}$ as well $A_{\text{no}}$ are called promise’s components.

*Corresponding author.

\textit{E-mail addresses:} zhengshenggen@gmail.com (S. Zheng), lilvzh@mail.sysu.edu.cn (L. Li), gruska@fi.muni.cz (J. Gruska).
The goal is then seen to decide, given a string $x$ from the promise set, whether $x \in A_{\text{yes}}$ or $x \in A_{\text{no}}$. In a special (trivial) case the promise is the whole set $\Sigma^\ast$. However, in general it may be very nontrivial to decided whether an input string is in the promise set.

In spite of the fact that both papers brought interesting outcomes and excellent demonstrations of both problems and results, the study of promise problems did not get a special momentum yet.

On the other side, very large impact had the results concerning several promise problems in quantum information processing. They demonstrated that using quantum phenomena and processes one can solve several interesting promise problems with much less quantum queries (to quantum black boxes) than in the case only classical tools and queries (to classical black boxes) are available. The initial developments in this area culminated by the result of Simon [31] that the promise problem he introduced can be solved with the polynomial number of quantum and classical queries but not with polynomial number of classical queries only even if probabilistic tools are used. The second promise problem that took very large attention, especially its special cases, for example integer factorization, due to Shor [30], can be now seen as one of the most fundamental, and still open, problems is that of the Hidden Subgroup Problem for non-commutative groups.

Almost all papers so far, especially from [13, 17] dealt with promise problems in the context of such high level complexity classes as $\text{P}$, $\text{NP}$, $\text{BPP}$, $\text{SZK}$ and so on.

In this paper we try to explore promise problems on another level. Namely using classical and quantum or even semi-quantum finite automata working in various (especially two special) modes. At first we deal with closure and ordering properties of promise problems. Afterwards, lower and upper bounds are derived concerning the state complexity in a promise problem between the promise and its two components. Finally, it is shown that quantum finite automata can be more powerful than classical finite automata in solving promise problems.

2. Preliminaries

We introduce in this section some basic concepts and notations concerning finite automata. For quantum information processing and more on (quantum and semi-quantum) finite automata we refer the reader to [18, 21, 28].

2.1. Deterministic finite automata

In this subsection we recall the definition of deterministic finite automata (DFAs) and give the definition of so-called promise version deterministic finite automata (PVDFAs).

**Definition 2.** A deterministic finite automaton (DFA) $\mathcal{A}$ is specified by a 5-tuple

$$\mathcal{A} = (S, \Sigma, \delta, s_0, S_a),$$

where:

- $S$ is a finite set of classical states;
- $\Sigma$ is a finite set of input symbols;
- $s_0 \in S$ is the initial state of the automaton;
\[ S_a \subseteq S \] is a set of accepting states;

\[ \delta \] is a transition function:
\[ \delta : S \times \Sigma \rightarrow S. \]  

(2)

For any \( w \in \Sigma^* \) and \( \sigma \in \Sigma \), we define
\[ \hat{\delta}(s, w\sigma) = \hat{\delta}(\hat{\delta}(s, w), \sigma) \]  

(3)

and if \( w \) is the empty string, then
\[ \hat{\delta}(s, w\sigma) = \delta(s, \sigma). \]  

(4)

A language \( L \) is recognized by a DFA \( \mathcal{A} \) if for every \( w \in \Sigma^* \)

- \( w \in L \) if and only if \( \hat{\delta}(s_0, w) \in S_a \).
- \( w \notin L \) if and only if \( \hat{\delta}(s_0, w) \notin S_a \).

It is well known that a language \( L \) is recognized by a DFA if and only if \( L \) is regular.

A promise problem \( \mathcal{A} = (A_{yes}, A_{no}) \) is solved by a DFA \( \mathcal{A} \) if for every \( w \in A_{yes} \cup A_{no} \)

- \( w \in A_{yes} \) implies that \( \hat{\delta}(s_0, w) \in S_a \).
- \( w \in A_{no} \) implies that \( \hat{\delta}(s_0, w) \notin S_a \).

**Definition 3.** A *promise version deterministic finite automaton* (pvDFA) \( \mathcal{A} \) is specified by a 6-tuple
\[ \mathcal{A} = (S, \Sigma, \delta, s_0, S_a, S_r), \]  

(5)

where \( S_a \) is a set of accepting states and \( S_r \) is a set of rejecting states, respectively.

If \( S_a \cup S_r = S \), then \( \mathcal{A} \) is a DFA. DFAs can therefore be considered as a special case of pvDFAs.

A promise problem \( \mathcal{A} = (A_{yes}, A_{no}) \) is recognized by a pvDFA \( \mathcal{A} \) if for every \( w \in \Sigma^* \)

- \( w \in A_{yes} \) if and only if \( \hat{\delta}(s_0, w) \in S_a \).
- \( w \in A_{no} \) if and only if \( \hat{\delta}(s_0, w) \in S_r \).

A promise problem \( \mathcal{A} = (A_{yes}, A_{no}) \) is solved by a pvDFA \( \mathcal{A} \) if for every \( w \in A_{yes} \cup A_{no} \)

- \( w \in A_{yes} \) implies that \( \hat{\delta}(s_0, w) \in S_a \).
- \( w \in A_{no} \) implies that \( \hat{\delta}(s_0, w) \in S_r \).

We will see that for pvDFAs recognizability and solvability aspects are different.
2.2. Quantum and semi-quantum finite automata—basic models and working modes

Quantum finite automata were first introduced by Kondacs and Watrous [23] and by Moore and Crutchfields [26]. It has been proved that one-way quantum finite automata (1QFAs) are not more powerful than one-way classical finite automata (1FAs) [2, 24]. However, 1QFA can have some state complexity advantages in recognizing languages or solving promise problems [2–4, 8–11, 14, 20, 33, 38].

Definition 4. A measure-once quantum finite automaton (MO-1QFA) \( M \) is specified by a 5-tuple

\[
M = (Q, \Sigma, \{ U_\sigma \mid \sigma \in \Sigma' \}, |0\rangle, Q_a)
\]

where:

- \( Q \) is a finite set of orthonormal quantum (basis) states, denoted as \( \{|i\rangle \mid 0 \leq i < |Q|\} \);
- \( \Sigma \) is a finite alphabet of input symbols and \( \Sigma' = \Sigma \cup \{\#, \$\} \) (where \( \$ \) will be used as the left end-marker and \( \# \) as the right end-marker);
- \( |0\rangle \in Q \) is the initial quantum state;
- \( Q_a \subseteq Q \) denotes the set of accepting basis states;
- \( U_\sigma \)'s (\( \sigma \in \Sigma' \)) are unitary operators.

The quantum state space of this model will be a \(|Q|\)-dimensional Hilbert space denoted \( \mathcal{H}_Q \). Each quantum basis state \( |i\rangle \) in \( \mathcal{H}_Q \) can be represented by a column vector with the \((i+1)\)th entry being 1 and other entries being 0. With this notational convenience we can describe the above model as follows:

1. The initial state \( |0\rangle \) is represented as \( |q_0\rangle = (1, 0, \ldots, 0)^T \).
2. The accepting set \( Q_a \) corresponds to the projective operator \( P_{acc} = \sum_{|i\rangle \in Q_a} |i\rangle \langle i| \).

The computation of an MO-1QFA \( M \) on an input string \( x = \sigma_1\sigma_2\cdots\sigma_n \in \Sigma^* \) goes as follows: \( M \) “reads” the input string from the left end-marker to the right end-marker, symbol by symbol, and the unitary matrices \( U_b, U_{\sigma_1}, U_{\sigma_2}, \ldots, U_{\sigma_n}, U_b \) are applied, one by one, always on the current state, starting with \( |0\rangle \) as the initial state. Finally, the projective measurement \( \{P_{acc}, I - P_{acc}\} \) is performed on the final state, in order to accept or reject the input. Therefore, for an input string \( w = \sigma_1\sigma_2\cdots\sigma_n \), \( M \) has the accepting probability

\[
Pr[M \text{ accepts } w] = \|P_{acc}U_bU_{\sigma_n}\cdots U_{\sigma_2}U_{\sigma_1}|0\rangle\|^2
\]

and the rejecting probability

\[
Pr[M \text{ rejects } w] = 1 - Pr[M \text{ accepts } w].
\]

Definition 5. A promise version measure-once quantum finite automaton (pvMO-1QFA) \( M \) is specified by a 6-tuple

\[
M = (Q, \Sigma, \{ U_\sigma \mid \sigma \in \Sigma' \}, |0\rangle, Q_a, Q_r)
\]

where: \( Q, \Sigma, \Sigma', |0\rangle, Q_a, U_\sigma \) are as defined in an MO-1QFA, \( Q_r \subseteq Q \) (\( Q_r \cap Q_a = \emptyset \)) denotes the set of rejecting basis states. The set \( Q_r \) corresponds to the projective operator \( P_{rej} = \sum_{|i\rangle \in Q_r} |i\rangle \langle i| \).
For an input string \( w = \sigma_1 \sigma_2 \cdots \sigma_n \), \( M \) has the accepting probability

\[
Pr[M \text{ accepts } w] = \| P_{acc} U_\sigma \cdots U_{\sigma_n} U_{\sigma_1} |0\rangle \|^2.
\]

The rejecting probability is then

\[
Pr[M \text{ rejects } w] = \| P_{rej} U_\sigma \cdots U_{\sigma_n} U_{\sigma_1} |0\rangle \|^2.
\]

Two-way finite automata with quantum and classical states (2QCFAs) were introduced by Ambainis and Watrous [1] and explored in [33, 35–38]. 1QCFAs are one-way versions of 2QCFAs, which were introduced by Zheng et al. [35]. Informally, a 1QCFA can be seen as a DFA which has access to a quantum memory of a constant size (dimension), upon which it performs quantum transformations and measurements. Given a finite set of quantum basis states \( Q \), we denote by \( H(Q) \) the Hilbert space spanned by \( Q \). Let \( U(H(Q)) \) and \( O(H(Q)) \) denote the sets of unitary operators and projective measurements over \( H(Q) \), respectively.

**Definition 6.** A one-way finite automaton with quantum and classical states (1QCF) \( A \) is specified by a 10-tuple

\[
A = (Q, S, \Sigma, \Theta, \Delta, \delta, |q_0\rangle, s_0, S_a, S_r)
\]

where:

1. \( Q \) is a finite set of orthonormal quantum basis states.
2. \( S \) is a finite set of classical states.
3. \( \Sigma \) is a finite alphabet of input symbols and \( \Sigma' = \Sigma \cup \{\$\} \), where \( \$ \) will be used as the left end-marker and \( \$ \) as the right end-marker.
4. \( |q_0\rangle \in Q \) is the initial quantum state.
5. \( s_0 \) is the initial classical state.
6. \( S_a \subset S \) and \( S_r \subset S \), where \( S_a \cap S_r = \emptyset \), are sets of the classical accepting and rejecting states, respectively.
7. \( \Theta \) is a quantum transition function

\[
\Theta : S \setminus (S_a \cup S_r) \times \Sigma' \to U(H(Q)),
\]

assigning to each pair \((s, \gamma)\) a unitary transformation.
8. \( \Delta \) is a mapping

\[
\Delta : S \times \Sigma' \to O(H(Q)),
\]

where each \( \Delta(s, \gamma) \) corresponds to a projective measurement (a projective measurement will be taken each time a unitary transformation is applied; if we do not need a measurement, we denote that \( \Delta(s, \gamma) = I \), and we assume the result of the measurement to be a fixed \( c \)).
9. \( \delta \) is a special transition function of classical states. Let the results set of the measurement be \( \mathcal{C} = \{c_1, c_2, \ldots, c_s\} \), then

\[
\delta : S \times \Sigma' \times \mathcal{C} \to S,
\]

where \( \delta(s, \gamma)(c_i) = s' \) means that if a tape symbol \( \gamma \in \Sigma' \) is being scanned and the projective measurement result is \( c_i \), then the state \( s \) is changed to \( s' \).
Given an input word $w = \sigma_1 \cdots \sigma_n$, the word on the tape will be seen as $w = \{w\}$ (for convenience, we denote $\sigma_0 = \epsilon$ and $\sigma_{n+1} = \$$. Now, we define the behavior of 1QCFA $M$ on the input word $w$. The computation starts in the classical state $s_0$ and the quantum state $|q_0\rangle$, then the transformations associated with symbols in the word $\sigma_0 \sigma_1 \cdots, \sigma_{n+1}$ are applied in succession. The transformation associated with a state $s \in S$ and a symbol $\sigma \in \Sigma'$ consists of three steps:

1. Firstly, $\Theta(s, \sigma)$ is applied to the current quantum state $|\phi\rangle$, yielding the new state $|\phi'\rangle = \Theta(s, \sigma)|\phi\rangle$.
2. Secondly, the observable $\Delta(s, \sigma) = \mathcal{O}$ is measured on $|\phi'\rangle$. The set of possible results is $\mathcal{C} = \{c_1, \cdots, c_k\}$.
3. Thirdly, the current classical state $s$ will be changed to $\delta(s, \sigma)(c_k) = s'$.

An input word $w$ is assumed to be accepted (rejected) if and only if the automaton enters an accepting (rejecting) state. We assume that $\delta$ is well defined so that 1QCFA $M$ always accepts or rejects at the end of the computation.

Let $0 \leq \varepsilon \leq \frac{1}{2}$. An MO-1QFA, 1QCFA $M$ recognizes a language $L$ with an error probability $\varepsilon$ if for every $w \in \Sigma^*$

- $w \in L$ if and only if $\Pr[M \text{ accepts } w] \geq 1 - \varepsilon$.
- $w \notin L$ if and only if $\Pr[M \text{ rejects } w] \geq 1 - \varepsilon$.

A pvMO-1QFA $M$ recognizes a promise problem $A = (A_{yes}, A_{no})$ with an error probability $\varepsilon$ if for every $w \in \Sigma^*$

- $w \in A_{yes}$ if and only if $\Pr[M \text{ accepts } w] \geq 1 - \varepsilon$.
- $w \in A_{no}$ if and only if $\Pr[M \text{ rejects } w] \geq 1 - \varepsilon$.

A promise problem $A = (A_{yes}, A_{no})$ is solved by a pvMO-1QFA, 1QCFA $M$ with an error probability $\varepsilon$ if for every $w \in A_{yes} \cup A_{no}$

- $w \in A_{yes}$ implies that $\Pr[M \text{ accepts } w] \geq 1 - \varepsilon$, and
- $w \in A_{no}$ implies that $\Pr[M \text{ rejects } w] \geq 1 - \varepsilon$.

If $\varepsilon = 0$, we say that the automaton $M$ solves (recognizes) the promise problem $A$ exactly.

If a promise problem $A$ is recognized by a promise version finite automaton (PVFA) $M$ with an error probability $\varepsilon$, it does not mean that the promise problem $A$ is recognized by a PVFA $M$ with an error probability $\varepsilon'$ such that $\varepsilon' > \varepsilon$. However, it does mean that the promise problem $A$ can be solved by the PVFA $M$ with an error probability $\varepsilon'$ such that $\varepsilon' \geq \varepsilon$. If a promise problem $B$ is solved by a PVFA $M$ with an error probability $\varepsilon$, it does not necessarily mean that $B$ is recognized by the PVFA $M$ with the error probability $\varepsilon$.

3. Properties of pvDFA

We discuss the properties of promise problems that can be recognized or solved by pvDFA.

**Theorem 1.** A promise problem $A = (A_{yes}, A_{no})$ can be recognized by a pvDFA $A$ iff $A_{yes}$ and $A_{no}$ are regular.
Proof. \((\Rightarrow)\) Suppose that a promise problem \(A\) can be recognized by a pvDFA \(\mathcal{A} = (S, \Sigma, \delta, s_0, S_\text{yes}, S_\text{no})\). In such a case, for all \(w \in \Sigma^*\), \(w \in A_{\text{yes}}\) if and only if \(\delta(s_0, w) \in S_\text{yes}\). Let DFA \(\mathcal{A}' = (S, \Sigma, \delta, s_0, S_\text{yes})\). Obviously, \(A_{\text{yes}}\) is recognized by \(\mathcal{A}'\) and therefore \(A_{\text{yes}}\) is regular. Using similar argument, we can show that \(A_{\text{no}}\) is regular.

\((\Leftarrow)\) We assume that \(A_{\text{yes}}\) can be recognized by a DFA \(\mathcal{A}^1 = (S^1, \Sigma, \delta^1, s_0^1, S_\text{yes}^1)\) and \(A_{\text{no}}\) can be recognized by a DFA \(\mathcal{A}^2 = (S^2, \Sigma, \delta^2, s_0^2, S_\text{no}^2)\). We now consider the pvDFA \(\mathcal{A} = (S, \Sigma, \delta, s_0, S_\text{yes}, S_\text{no})\) where

\[ S = (S^1 \times S^2) \setminus (S_\text{yes}^1 \times S_\text{no}^2); \]

\[ s_0 = \langle s_0^1, s_0^2 \rangle; \]

\[ \delta(\langle s^1, s^2 \rangle, \sigma) = \langle \delta^1(s^1, \sigma), \delta^2(s^2, \sigma) \rangle; \]

\[ S_\text{yes} = S^1 \times (S^2 \setminus S^2_\text{no}) \text{ and } S_\text{no} = (S^1 \setminus S^1_\text{yes}) \times S^2_\text{no}. \]

For any \(w \in \Sigma^*\), we prove that \(s = \widehat{\delta}(s_0, w) \notin S_\text{yes} \times S_\text{no}\) first. Let us assume that \(s = \langle s^1, s^2 \rangle \in S^1 \times S^2\). We have \(\widehat{\delta}(s_0, w) = \langle \widehat{\delta}^1(s_0^1, w), \widehat{\delta}^2(s_0^2, w) \rangle = \langle s^1, s^2 \rangle\). Therefore \(\widehat{\delta}^1(s_0^1, w) = s_1 \in S^1_\text{yes}\) and \(\widehat{\delta}^2(s_0^2, w) = s_2 \in S^2_\text{no}\). Thus implies that \(w \in A_{\text{yes}}\) and \(w \in A_{\text{no}}\), which is a contradiction.

If \(w \in A_{\text{yes}}\), then \(s^1 = \widehat{\delta}^1(s_0^1, w) \in S^1_\text{yes}\) and \(s^2 = \widehat{\delta}^2(s_0^2, w) \notin S^2_\text{no}\). Therefore, \(\widehat{\delta}(s_0, w) = \langle \widehat{\delta}^1(s_0^1, w), \widehat{\delta}^2(s_0^2, w) \rangle = \langle s^1, s^2 \rangle \in S^1 \times (S^2 \setminus S^2_\text{no}) = S_\text{yes}\).

If \(w \in \Sigma^*\) such that \(\widehat{\delta}(s_0, w) \in S_\text{no}\), then \(\widehat{\delta}(s_0, w) = \langle \widehat{\delta}^1(s_0^1, w), \widehat{\delta}^2(s_0^2, w) \rangle = \langle s^1, s^2 \rangle \in S^1 \times (S^2 \setminus S^2_\text{no})\). We have therefore \(\widehat{\delta}^1(s_0^1, w) \in S^1_\text{yes}\) and \(w \in A_{\text{yes}}\).

With a similar argument as above, we can get that for any \(w \in \Sigma^*\), \(w \in A_{\text{no}}\) if and only if \(\widehat{\delta}(s_0, w) \in S_\text{no}\). Therefore the promise problem \(A = (A_{\text{yes}}, A_{\text{no}})\) can be recognized by the pvDFA \(\mathcal{A}\).

**Remark 1.** If a promise problem \(A\) is recognized by a pvDFA \(\mathcal{A}\), then \(A\) is solved by the same pvDFA \(\mathcal{A}\). However, if a promise problem \(A\) is solved by a pvDFA \(\mathcal{A}\), it does not necessarily mean that the promise problem \(A\) can be recognized by a pvDFA. For example, we consider a promise problems \(B^l = (B^l_{\text{yes}}, B^l_{\text{no}})\) with \(B^l_{\text{yes}} = \{a^ib^i \mid i \geq 0\}\) and \(B^l_{\text{no}} = \{a^ib^{i+1} \mid i \geq 0\}\), where \(l\) is a fix positive integer. The promise problem \(B^l\) can be solved by a DFA, therefore it can be solved by a pvDFA. However, both \(B^l_{\text{yes}}\) and \(B^l_{\text{no}}\) are nonregular languages. Therefore \(B^l\) cannot be recognized by a pvDFA.

### 3.1. Pumping Lemmas

The pumping lemma for pvDFA concerning recognition is similar to the classical one [21].

**Lemma 1 (Pumping Lemma I).** Let a promise problem \(A = (A_{\text{yes}}, A_{\text{no}})\) be recognized by a pvDFA \(\mathcal{A}\). Then there exists an integer \(p \geq 1\), depending only on \(\mathcal{A}\), such that every string \(w\) in \(A_{\text{yes}}\) (\(A_{\text{no}}\)) of length at least \(p\) can be written as \(w = xyz\) (i.e., \(w\) can be divided into three substrings), satisfying the following conditions:

- \(|y| \geq 1;\)
- \(|xy| \leq p;\)
- \(xyz^t \in A_{\text{yes}}\) (\(A_{\text{no}}\) for all integers \(t \geq 0\)).

Now we consider a pumping lemma for pvDFA concerning solvability.
Lemma 2 (Pumping Lemma II). Let a promise problem \( A = (A_{yes}, A_{no}) \) can be solved by a pvDFA \( A \). Then there exists an integer \( p \geq 1 \), depending only on \( A \), such that every string \( w \) in \( A_{yes} (A_{no}) \) of length at least \( p \) can be written as \( w = xyz \) (i.e., \( w \) can be divided into three substrings), satisfying the following conditions:

- \( |y| \geq 1 \);
- \( |xy| \leq p \);
- \( xy^tz \notin A_{no} (A_{yes}) \) for all integers \( t \geq 0 \).

Proof. Let the pvDFA \( A = (S, \Sigma, \delta, s_0, S_a, S_r) \) and \( p = |S| \) be the number of states. For a word \( w = \sigma_1 \ldots \sigma_n \in A_{yes} (A_{no}) \), the transitions of \( A \) on \( w \) is \( \delta(s_{i-1}, \sigma_i) = s_i \) for \( 0 \leq i \leq n \) and \( s_n \in S_a (S_r) \).

If \( n \geq p \), then there exist \( i < j \) such that \( s_i = s_j \). Let \( x = \sigma_1 \ldots \sigma_i, y = \sigma_{i+1} \ldots \sigma_j \) and \( z = \sigma_{j+1} \ldots \sigma_n \). We have, \( \hat{\delta}(s_0, x) = s_i, \hat{\delta}(s_i, y) = s_j \) and \( \hat{\delta}(s_j, z) = s_n \in S_a \). Therefore \( \hat{\delta}(s_i, y^tz) = s_i \).

If there exists an integer \( t \geq 0 \) such that \( w = xy^tz \in A_{no} (A_{yes}) \), then \( \hat{\delta}(s_0, w) = \hat{\delta}(s_0, xy^tz) = \hat{\delta}(s_i, y^tz) = \hat{\delta}(s_i, z) = s_n \in S_a (S_r) \), which is a contradiction. Therefore, we have \( xy^tz \notin A_{no} (A_{yes}) \) for all \( t \geq 0 \).

3.2 Closure properties

Let us have promise problems \( A = (A_{yes}, A_{no}) \), \( B = (B_{yes}, B_{no}) \) and \( C = (C_{yes}, C_{no}) \). The complement, intersection and union operations on promise problems are defined as follows.

- Complement: \( \overline{A} = (\overline{A_{yes}}, \overline{A_{no}}) \), where \( \overline{A_{yes}} = A_{no} \) and \( \overline{A_{no}} = A_{yes} \).
- Intersection: \( C = A \cap B = (C_{yes}, C_{no}) \), where \( C_{yes} = A_{yes} \cap B_{yes} \) and \( C_{no} = A_{no} \cap B_{no} \).
- Union: if \( (A_{yes} \cup B_{yes}) \cap (A_{no} \cup B_{no}) \neq \emptyset \), then the union of \( A \) and \( B \) is undefined; otherwise the union \( C = A \cup B = (C_{yes}, C_{no}) \), where \( C_{yes} = A_{yes} \cup B_{yes} \) and \( C_{no} = A_{no} \cup B_{no} \).

Theorem 2. If a promise problem \( A \) can be recognized (solved) by a pvDFA \( A \), then \( \overline{A} \) can be recognized (solved) by a pvDFA.

Proof. Suppose that the promise problem \( A \) can be recognized (solved) by a pvDFA \( A = (S, \Sigma, \delta, s_0, S_a, S_r) \). Exchanging the sets of accepting states and rejecting states of the pvDFA \( A \), we now consider a new pvDFA \( A' = (S, \Sigma, \delta, s_0, S_r, S_a) \). It is easy to prove that \( \overline{A} \) is recognized (solved) by the pvDFA \( A' \).

Theorem 3. If promise problems \( A \) and \( B \) can be recognized by pvDFAs, then their intersection can be also recognized by a pvDFA.

Proof. Suppose that the promise problem \( A \) can be recognized by pvDFA \( A^1 = (S^1, \Sigma^1, \delta^1, s^1_0, S^1_a, S^1_r) \) and the promise problem \( B \) can be recognized by pvDFA \( A^2 = (S^2, \Sigma^2, \delta^2, s^2_0, S^2_a, S^2_r) \). We consider a pvDFA \( A = (S, \Sigma, \delta, s_0, S_a, S_r) \), where

- \( S = S^1 \times S^2 \);
- \( \Sigma = \Sigma^1 \cap \Sigma^2 \);
- \( s_0 = \langle s^1_0, s^2_0 \rangle \);
• \( \delta((s^1, s^2), \sigma) = (\delta^1(s^1, \sigma), \delta^2(s^2, \sigma)) \);

• \( S_a = S_a^1 \times S_a^2 \) and \( S_r = S_r^1 \times S_r^2 \).

Let the promise problem \( C = (C_{yes}, C_{no}) \) be the union of the promise problems \( A = (A_{yes}, A_{no}) \) and \( B = (B_{yes}, B_{no}) \).

If \( w \in C_{yes} \), then \( w \in A_{yes} \cap B_{yes} \). We have \( \hat{\delta}(s_0, w) = \hat{\delta}(s_0^1, w) \in S_a^1 \) and \( \hat{\delta}(s_0^2, w) \in S_a^2 \). Therefore we have \( \hat{\delta}(s_0, w) = (\hat{\delta}(s_0^1, w), \hat{\delta}(s_0^2, w)) \in S_a^1 \times S_a^2 = S_a \).

If \( w \in \Sigma^* \) such that \( \hat{\delta}(s_0, w) \in S_a \), we have \( \delta(s_0, w) = (\delta(s_0^1, w), \delta(s_0^2, w)) \in S_a = S_a^1 \times S_a^2 \).

Therefore, \( s_0, w \in S_a^1 \) and \( s_2, \delta(s_0^2, w) \in S_a^2 \), i.e. \( w \in A_{yes} \) and \( w \in B_{yes} \). Hence, \( w \in A_{yes} \cap B_{yes} = C_{yes} \).

Therefore, we have \( w \in C_{yes} \) if and only if \( \hat{\delta}(s_0, w) \in S_a \). By a similar argument, we can have \( w \in C_{no} \) if and only if \( \hat{\delta}(s_0, w) \in S_r \). Hence, the promise problem \( C = A \cap B \) can be recognized by the pvDFA \( \mathcal{A} \).

**Theorem 4.** If promise problems \( A \) and \( B \) can be solved by pvDFAs, then their intersection can be solved also by a pvDFA.

**Proof.** Let promise problem \( C = (C_{yes}, C_{no}) \) be the union of the promise problems \( A = (A_{yes}, A_{no}) \) and \( B = (B_{yes}, B_{no}) \). Suppose that the promise problem \( A = (A_{yes}, A_{no}) \) can be solved by pvDFA \( \mathcal{A} \). Since \( C_{yes} = A_{yes} \cap B_{yes} \subset A_{yes} \) and \( C_{no} = A_{no} \cap B_{no} \subset A_{no} \), the promise problem \( C \) can be solved by \( \mathcal{A} \).

**Theorem 5.** If promise problems \( A \) and \( B \) can be recognized by pvDFA and their union \( C \) exists, then \( C \) can be recognized also by a pvDFA.

**Proof.** If \( \Sigma^1 \neq \Sigma^2 \), then we let \( \Sigma = \Sigma^1 \cup \Sigma^2 \). It is easy to extend the pvDFA to recognize the promise problems \( A(B) \) with alphabet \( \Sigma \) by adding one state to the original pvDFA. Now suppose that the promise problem \( A \) with the alphabet \( \Sigma \) can be recognized by a pvDFA \( \mathcal{A}^1 = (S^1, \Sigma, \delta^1, s_0^1, S^1_a, S^1_r) \) and the promise problem \( B \) with alphabet \( \Sigma \) can be recognized by a pvDFA \( \mathcal{A}^2 = (S^2, \Sigma, \delta^2, s_0^2, S^2_a, S^2_r) \).

We consider a ZDFA \( \mathcal{A} = (S, \Sigma, \delta, s_0, S_a, S_r) \), where

• \( S = (S_1 \times S_2) \setminus ((S_a^1 \times S_a^2) \cup (S_r^1 \times S_r^2)) \);

• \( \Sigma = \Sigma^1 \cup \Sigma^2 \);

• \( s_0 = (s_0^1, s_0^2) \);

• \( \delta((s^1, s^2), \sigma) = (\delta^1(s^1, \sigma), \delta^2(s^2, \sigma)) \);

• \( S_a = \{ (s^1, s^2) | s^1 \in S_a^1 \text{ or } s^2 \in S_a^2 \} \) and \( S_r = \{ (s^1, s^2) | s^1 \in S_r^1 \text{ or } s^2 \in S_r^2 \} \).

Let the promise problem \( C = (C_{yes}, C_{no}) \) be the union of promise problems \( A = (A_{yes}, A_{no}) \) and \( B = (B_{yes}, B_{no}) \). Since the union \( C = A \cup B \) exists, we have \( (A_{yes} \cup B_{yes}) \cap (A_{no} \cup B_{no}) = \emptyset \).

For any \( w \in \Sigma^* \), we prove that \( s = \hat{\delta}(s_0, w) \notin S_a^1 \times S_a^2 \) first. We assume that \( s = (s^1, s^2) \in S_a^1 \times S_a^2 \). We have \( \delta(s_0, w) = (\delta^1(s_0^1, w), \delta^2(s_0^2, w)) \in (s^1, s^2) \). Therefore \( \hat{\delta}(s_0^1, w) = s_1 \in S_a^1 \) and \( \hat{\delta}(s_0^2, w) = s_2 \in S_a^2 \). It follows that \( w \in A_{yes} \) and \( w \in B_{no} \). Therefore \( w \in (A_{yes} \cup B_{yes}) \cap (A_{no} \cup B_{no}) = \emptyset \), which is a contradiction. By a similar argument, we can prove that \( \hat{\delta}(s_0, w) \notin S_r^1 \times S_r^2 \). Hence \( S_a \cap S_r = \emptyset \).

If \( w \in C_{yes} \), then \( w \in A_{yes} \cup B_{yes} \). We have \( \hat{\delta}(s_0^1, w) \in S_a^1 \) or \( \hat{\delta}(s_0^2, w) \in S_a^2 \). Therefore \( \hat{\delta}(s_0, w) = \hat{\delta}(s_0^1, w) \in S_a^1 \in S_a \).
If \( w \in \Sigma^* \) is such that \( \delta(s_0, w) \in S_a \), we have \( \delta(s_0, w) = \delta((s_0', s_0''), w) = (\delta s_1(w), \delta s_2(w)) \in S_a \). Therefore \( \delta s_1(w), \delta s_2(w) \in S_a \) and \( \delta s_1(w) = S_a \), i.e. \( w \in A_{yes} \) or \( w \in B_{yes} \). Hence, \( w \in A_{yes} \cup B_{yes} = C_{yes} \).

Therefore, \( w \in C_{yes} \) if and only if \( \delta(s_0, w) \in S_a \). By a similar argument, we can show that \( w \in C_{no} \) if and only if \( \delta(s_0, w) \in S_r \). Hence, the promise problem \( C = A \cup B \) can be recognized by the pvDFA \( A \).

**Remark 2.** If promise problems \( A \) and \( B \) can be solved by pvDFAs and their union \( C \) exists, then \( C \) may not be solved by a pvDFA. Indeed, let \( A = (A_{yes}, A_{no}), \) where \( A_{yes} = \{a^nb^n \mid n \text{ is odd} \} \) and \( A_{no} = \{a^nb^m \mid m \neq n \} \) and at least one of \( m, n \) is even}. If \( w \in A_{yes} \), then \( \#(w) = 2 \) and \( \#(w) \) are odd. If \( w \in A_{no} \), at least one of \( \#(w) \) is even. Obviously, we can design a pvDFA to solve the promise problem \( A \). Let \( B = (B_{yes}, B_{no}) \), where \( B_{yes} = \{a^nb^n \mid n \text{ is even} \} \) and \( A_{no} = \{a^nb^m \mid m \neq n \} \) and at least one of \( m, n \) is odd}. Similarly, we can design another pvDFA to solve the promise problem \( B \). Now we consider their union \( C = A \cup B = (C_{yes}, C_{no}), \) where \( C_{yes} = A_{yes} \cup B_{yes} = \{a^nb^n \} \) and \( C_{no} = A_{no} \cup B_{no} = \{a^nb^m \mid n \neq m \} \).

We use the Pumping Lemma II to prove that the promise problem \( C \) can not be solved by anypvDFA. Assume on contrary that \( C \) can be solved by a pvDFA \( A \) and \( p \) is the constant for the pumping lemma. Choose \( w = a^pb^p \in A_{yes} \). Clearly, \( |w| > p \). By the pumping lemma, \( w = xyz \) for some \( x, y, z \in \Sigma^* \) such that (1) \( |y| \leq p \), (2) \( |y| \geq 1 \), and (3) \( xy^t \notin A_{no} \) for all \( t \geq 0 \). By (1) and (2), we have \( y = a^k, 1 \leq k \leq p \). However, \( xy^t \in A_{no} \). Therefore, (3) does not hold. The promise problem \( C \) therefore does not satisfy the pumping property of the Pumping Lemma II. Hence, the promise problem \( C \) can not be solved by any pvDFA.

### 3.3. Ordering of promise problems

Let \( A = (A_{yes}, A_{no}) \) and \( B = (B_{yes}, B_{no}) \) be two promise problems over \( \Sigma \). We say \( A \) is equivalent to \( B \), denoted by \( A = B \), if \( A_{yes} = B_{yes} \) and \( A_{no} = B_{no} \). We say that \( A \) is a subproblem of \( B \), denoted by \( C < A \), if \( A_{yes} \subseteq B_{yes} \) and \( A_{no} \subseteq B_{no} \) and \( A \) is not equivalent to \( B \).

Let \( A \) and \( B \) be two promise problems over \( \Sigma \) recognized by pvDFA \( A \) and \( B \), respectively. We say \( \text{pvDFA } A \) is equivalent to \( \text{pvDFA } B \) (denoted by \( A = B \)) if \( A = B \). We say pfDFA \( B \) is more powerful than pfDFA \( A \) (denoted by \( B > A \) or \( A < B \) ) if \( A < B \).

**Theorem 6.** If a promise problem \( A \) can be solved by pvDFA \( A \) and \( A < B \), then the promise problem \( A \) can be solved by pvDFA \( B \).

**Proof.** Obvious.

**Remark 3.** From the above theorem, we can see that if \( \text{pvDFA } A < B \), then we can use pvDFA \( B \) to replace pvDFA \( A \) in solving promise problems without changing pvDFA \( B \). However, we cannot use \( \text{DFA } B \) to replace \( \text{DFA } A \) in solving promise problems unless \( B = A \).

We say a pvDFA \( A \) is maximally powerful if there does not exist a pvDFA \( B \) such that \( A < B \).

**Theorem 7.** A pvDFA \( A \) is maximally powerful if and only if it is a DFA.

**Proof.** Suppose that \( A = (A_{yes}, A_{no}) \) is the promise problem recognized by \( A \).

If \( A \) is a DFA, then \( A_{yes} = \Sigma^* \setminus A_{no} \). Therefore, there does not exist a promise problem such that \( A < B \). Therefore, there does exist a pvDFA \( B \) such that \( A < B \), i.e. \( A \) is maximally powerful.

Assume that \( \text{pvDFA } A \) is maximally powerful and \( A \) is not a DFA. Suppose that the pvDFA \( A = (S, \Sigma, \delta, s_0, S_a, S_r) \). We have \( S_a \cup S_r \neq S \) and \( S_r \) is a proper subset of \( S \setminus S_a \). We now consider a new
pvDFA $B = (S, \Sigma, \delta, s_0, S_a, S \setminus S_a)$. Suppose that $B = (B_{yes}, B_{no})$ is the promise problem recognized by $B$. Therefore, there must exist some $w \in B_{no}$ such that $\delta(s_0, w) \in S \setminus S_a$ and $\delta(s_0, w) \notin S_r$. Therefore, $A_{no}$ is a propose subset of $B_{no}$. Since $A_{yes} = B_{yes}$. We have $A < B$ and therefore $A < B$, which is a contradiction. Hence, $A$ must be a DFA.

We say two pvDFAs $A$ and $B$ are comparable if $A = B$ or $A < B$ or $A > B$. DFAs are special case of pvDFAs. Two DFAs are either equivalent or not comparable. If pvDFA $A$ is a DFA, then there do not exist pvDFA $B$ such that $A < B$. Equivalence of two DFAs can be seen as a subcase of the equivalence of two pvDFAs.

4. State complexity

State complexity is one of the interesting topics in finite automata [34]. We now discuss the state complexity of pvDFAs for promise problems concerning recognizability and solvability.

For a regular language $L$, we denote by $s(L)$ the number of states in a minimal DFA to recognize language $L$. For a promise problem $A = (A_{yes}, A_{no})$ that can be recognized by pvDFA, we denote by $sr(A)$ the number of states in the minimal pvDFA to recognize $A$. For promise problem $A = (A_{yes}, A_{no})$ that can be solved by pvDFA, we denote by $ss(A)$ the number of states in the minimal pvDFA to solve $A$.

In a DFA $A = (S, \Sigma, \delta, s_0, S_a)$, a state $s$ is distinguishable from a state $t$ if there is at least one string $w \in \Sigma^*$ such that one of $\delta(s, w)$ and $\delta(t, w)$ is accepting, and the other is not accepting state. If every two states in DFA $A$ are distinguishable from each other, then $A$ is minimal [21].

**Theorem 8.** If a promise problem $A = (A_{yes}, A_{no})$ with $A_{yes} \neq \emptyset$ and $A_{no} \neq \emptyset$ can be recognized by a pvDFA, then

$$\max \{s(A_{yes}), s(A_{no})\} \leq sr(A) \leq s(A_{yes})s(A_{no}) - 1.$$  \hspace{1cm} (16)

**Proof.** Since $A$ can be recognized by a pvDFA, according to Theorem 1, $A_{yes}$ and $A_{no}$ are regular languages.

Suppose that $A$ is recognized by a minimal pvDFA $A_1 = (S, \Sigma, \delta, s_0, S_a)$ and the regular language $A_{yes}$ is recognized by the DFA $A_2 = (S, \Sigma, \delta, s_0, S_a)$, where $|S| \geq s(A_{yes})$ and $|S| > s(A_{no})$. Hence $sr(A) = |S| \geq \max \{s(A_{yes}), s(A_{no})\}$.

We assume that $A_{yes}$ is recognized by the minimal DFA $A^1 = (S^1, \Sigma, \delta^1, s_0^1, S_a^1)$ and $A_{no}$ is recognized by the minimal DFA $A^2 = (S^2, \Sigma, \delta^2, s_0^2, S_a^2)$. According to Theorem 1, the promise problem can be recognized by pvDFA $A = (S, \Sigma, \delta, s_0, S_a, S_r)$ where $S = (S^1 \times S^2) \setminus (S_a^1 \times S_a^2)$, $s_0 = \langle s_0^1, s_0^2 \rangle$, $\delta((s^1, s^2), \sigma) = (\delta^1(s^1, \sigma), \delta^2(s^2, \sigma))$, $S_a = S_a^1 \times (S^2 \setminus S_a^2)$ and $S_r = (S^1 \setminus S_a^1) \times S_a^2$. Therefore we have $sr(A) \leq |S| - |S_a^1 \times S_a^2| \leq S_1 \times S_2 - 1 = s(A_{yes})s(A_{no}) - 1$.

**Theorem 9.** The left side of the Inequality \((16)\) is tight.

**Proof.** We prove that $sr(A) = \max \{s(A_{yes}), s(A_{no})\}$ for some cases. Let us consider the promise problem $A^N, l = (A_{yes}^N, l, A_{no}^N, l)$ with $A_{yes}^N, l = \{a^i \mid i \geq 0\}$ and $A_{no}^N, l = \{a^{N+1} \mid i \geq 0\}$, where $N$ is a fix prime and $l$ is a positive integer such that $0 < l < N$. It is easy to see that $s(A_{yes}^N, l) = N$ and $s(A_{no}^N, l) = N$. Let us consider an $N$-state pvDFA $B = (S, \{a\}, \delta, s_0, S_a, S_r)$, where $S = \{s_0, s_1, \ldots, s_{N-1}\}$, $S_a = \{s_0\}$, $S_r = \{s_l\}$ and $\delta(s_i, a) = s_{(i+1) \mod N}$. It is easy to check that the promise problem $A^N, l$ can be recognized by the pvDFA $B$. 


Assume that the promise problem $A^{N,1}$ can be recognized an $M$-state pvDFA $B' = (S', \{a\}, \delta', s'_0, S'_a, S'_r)$ and $M < N$. It is easy to see that DFA $B'_1 = (S', \{a\}, \delta', s'_0, S'_a)$ can solve the promise problem $A^{N,1}$. Therefore, the minimal DFA to solve the promise problem $A^{N,1}$ has less than $N$ states, which contradicts to the fact that the minimal DFA to solve $A^{N,1}$ has $N$ states.

Therefore, $sr(A^{N,1}) = \max(s(A_{yes}), s(A_{no})) = N$. □

**Theorem 10.** The right side of the Inequality is tight up to some multiplication constant.

**Proof.** We prove that $sr(A) = \frac{1}{2} s(A_{yes}) s(A_{no})$ for some cases as follows.

Let $A_{yes} = \{(ap)^*\}$ and $A_{no} = \{(ap)^a\}$ where $p, q > 2$ are such that the gcd($p, q$) = 2. We first prove that $A_{yes} \cap A_{no} = \emptyset$. Since gcd($p, q$) = 2, there exist integers $k_1$ and $k_2$ such that $p = 2k_1$ and $q = 2k_2$. Assume that $A_{yes} \cap A_{no} \neq \emptyset$. There must exist integers $i$ and $j$ such that $(ap)^i = (ap)^j a$, i.e. $ip = jq + 1$. We have $1 = ip - jq = i(2k_1 - j2k_2) = 2(ik_1 - jk_2)$, which is a contradiction. Therefore $A_{yes} \cap A_{no} = \emptyset$. We consider now the promise problem $A = (A_{yes}, A_{no})$.

Since $A_{yes}$ and $A_{no}$ are regular languages, the promise problem $A$ can be recognized by a pvDFA. Let us consider the following pvDFA $A' = (S, \{a\}, \delta, s_0, S_a, S_r)$, where

1. $S = \{\langle s_k \mod p, s_k^2 \mod q \rangle | k \geq 0\}$;
2. $s_0 = \langle s_0^1, s_0^2 \rangle$;
3. $\delta(\langle s_k^1, s_k^2 \rangle, a) = \langle s_{k+1}^1 \mod p, s_{k+1}^2 \mod q \rangle$;
4. $S_a = \{\langle s_k^1 \mod p, s_k^2 \mod q \rangle | k \equiv 0 \mod p\}$ and $S_r = \{\langle s_k^1 \mod p, s_k^2 \mod q \rangle | k \equiv 1 \mod q\}$.

Firstly, we prove that $|S| = \frac{1}{2} pq$. Assume that there exist $0 \leq k_1 < k_2 < \frac{1}{2} pq - 1$ such that $\langle s_{k_1} \mod p, s_{k_1}^2 \mod q \rangle = \langle s_{k_2} \mod p, s_{k_2}^2 \mod q \rangle$. We have $k_1 < k_2 \mod p$ and $k_1 < k_2 \mod q$. Therefore $p|k(k_2 - k_1)$ and $q|k(k_2 - k_1)$. Since gcd($p, q$) = 2, we have $\frac{1}{2} pq(k_2 - k_1)$, which is a contradiction. Hence, $|S| \geq \frac{1}{2} pq$. For any $h \geq \frac{1}{2} pq$, let $h = i \times \frac{1}{2} pq + k$ where $0 \leq k < \frac{1}{2} pq$. Since $pq | \frac{1}{2} pq$ and $q | \frac{1}{2} pq$, we have $\langle s_h \mod p, s_h^2 \mod q \rangle = \langle s_k \mod p, s_k^2 \mod q \rangle$. Therefore $|S| = \frac{1}{2} pq$.

Secondly, we prove that $S_a \cap S_r = \emptyset$. Since gcd($p, q$) = 2, we have $2|p$ and $2|q$. Assume that $S_a \cap S_r \neq \emptyset$. There must exist integers $i$ and $j$ such that $k = ip$ and $k = jq + 1$. We have $ip = jq + 1$ and $2(ip - jq) = 1$, which is a contradiction.

Thirdly, it is easy to verify that the promise problem $A$ can be recognized by the pvDFA $A$. Therefore $sr(A) \leq |S| = \frac{1}{2} pq$.

Finally, we prove that the pvDFA $A' = (S, \{a\}, \delta, s_0, S_a, S_r)$ is minimal. Let us consider DFA $A' = (S, \{a\}, \delta, s_0, S_a)$. Obviously, the DFA $A'$ recognizes the language $A_{yes} \cup A_{no}$. We prove now that the DFA $A'$ is minimal. Let $F = S_a \cup S_r$ and $n = \frac{1}{2} pq$. For any $0 \leq i < j < n$, we prove that the states $s_i = \langle s_i^1 \mod p, s_i^2 \mod q \rangle$ and $s_j = \langle s_j^1 \mod p, s_j^2 \mod q \rangle$ are distinguishable. Since $s_i \neq s_j$, at most one of the following two conditions $(1) j - i \equiv 0 \mod p$ and $(2) j - i \equiv 0 \mod q$ holds. We have the following three cases:

1. Condition $1$ holds and $2$ does not hold. We have $\delta(s_i, a^{n-i+1}) = \langle s_{i+1}^1 \mod p, s_{i+1}^2 \mod q \rangle = \langle s_1^1, s_1^2 \rangle \notin F$ and $\delta(s_j, a^{n-i+1}) = \langle s_{j+1}^1 \mod p, s_{j+1}^2 \mod q \rangle = \langle s_1^1, s_1^2 \rangle \notin F$. Since $j - i \not\equiv 0 \mod q$, we have $j - i + 1 \equiv 0 \mod q$. Therefore $\delta(s_i, a^{n-i+1}) \notin F$. Hence $s_i$ and $s_j$ are distinguishable.

2. Condition $(2)$ holds and $(1)$ does not hold. The proof is similar to the one in the case $1$.

3. Neither condition $(1)$ nor $(2)$ holds. We have $\delta(s_i, a^{n-i}) = \langle s_{i}^1 \mod p, s_{i}^2 \mod q \rangle = \langle s_0^1, s_0^2 \rangle \notin F$ and $\delta(s_j, a^{n-i}) = \langle s_{j}^1 \mod p, s_{j}^2 \mod q \rangle = \langle s_{j-i}^1 \mod p, s_{j-i}^2 \mod q \rangle \notin F$. If $\delta(s_j, a^{n-i}) \notin F$, then $s_i$ and

12
We have shown that the DFA $\mathcal{A}'$ is minimal and it has $\frac{1}{2}pq$ states. Assume that there is a pvDFA $W$ with less than $\frac{1}{2}pq$ states recognizing the language $A_{yes} \cup A_{no}$. It then follows that the DFA $\mathcal{A}'$ is not minimal. A contradiction.

Obviously, $s(A_{yes}) = p$ and $s(A_{no}) = q$. We have proved that $sr(A) = \frac{1}{2}pq = \frac{1}{2}s(A_{yes})s(A_{no})$.

**Theorem 11.** If promise problem $A = (A_{yes}, A_{no})$ can be recognized by a pvDFA, then $ss(A) \leq \min\{s(A_{yes}), s(A_{no})\}$.

**Proof.** According to Theorem 11, $A_{yes}$ and $A_{no}$ are regular languages. Suppose $A_{yes}$ can be recognized by the minimal DFA $\mathcal{A}' = (S', \Sigma', \delta', s_0', S_2')$. This implies that the promise problem $A$ can be solved by the DFA $\mathcal{A}'$ and therefore $ss(A) \leq s(A_{yes})$. Suppose $A_{no}$ can be recognized by the minimal DFA $\mathcal{A}^2 = (S^2, \Sigma^2, \delta^2, s_0^2, S_2^2)$. We get that the promise problem $A$ can be solved by the DFA $\mathcal{A}^2$ and therefore $ss(A) \leq s(A_{no})$. Hence $ss(A) \leq \min\{s(A_{yes}), s(A_{no})\}$.

We prove that $ss(A) = \min\{s(A_{yes}), s(A_{no})\}$ for some cases. Let us consider the promise problem $A^{N, l} = (A_{yes}^{N, l}, A_{no}^{N, l})$ with $A_{yes}^{N, l} = \{a^N | i \geq 0\}$ and $A_{no}^{N, l} = \{a^{N+i} | i \geq 0\}$, where $N$ is a fix prime and $l$ is a positive integer such that $0 < l < N$. It is easy to see that $s(A_{yes}^{N, l}) = N$ and $s(A_{no}^{N, l}) = N$. It has been proved in [20] that $ss(A^{N, l}) = N$. Therefore $ss(A^{N, l}) = \min\{s(A_{yes}^{N, l}), s(A_{no}^{N, l})\} = N$.

**Remark 4.** $ss(A)$ can be very small. For example, let us consider the promise problem $A^{N, l} = (A_{yes}^{N, l}, A_{no}^{N, l})$ with $A_{yes}^{N, l} = \{a^N | i \geq 0\}$ and $A_{no}^{N, l} = \{a^{N+i} | i \geq 0\}$, where $N$ is a fix even integer and $l$ is a positive odd integer such that $0 < l < N$. Obviously, we have $s(A_{yes}^{N, l}) = N$ and $s(A_{no}^{N, l}) = N$. However $ss(A^{N, l}) = 2$, since the length of the input $|w|$ is even if $w \in A_{yes}^{N, l}$ and the length of the input $|w|$ is odd if $w \in A_{no}^{N, l}$.

5. Quantum finite automata advantages for promise problems

Recently, exact quantum computing has attracted more and more attentions. We prove now that exact quantum finite automata have advantages in recognizing promise problems comparing to their classical counterparts.

Let us consider a family of promise problems $A^l = (A_{yes}^l, A_{no}^l)$ with $A_{yes}^l = \{w \in \{a, b\}^* | \#_a(w) = \#_i(w)\}$ and $A_{no}^l = \{w \in \{a, b\}^* | \#_a(w) + l = \#_i(w)\}$, where $l$ is a fix positive integer such that $(2\pi i + \frac{\pi}{2}) \leq \sqrt{2l} \leq (2\pi i + \frac{3\pi}{2})$ for some integer $i$.

**Theorem 12.** The promise problems $A^l$ can be recognized exactly by a pvMO-1QFA and can not be recognized by any pvDFA.

**Proof.** Let $\theta = \sqrt{2\pi}, \, p = \cos \theta, \, \alpha = \sqrt{\frac{1}{1-p^2}} = \sqrt{\frac{\cos \theta}{1-\cos \theta}}$ and $\beta = \sqrt{\frac{1}{1-p^2}} = \sqrt{\frac{\cos \theta}{1-\cos \theta}}$. We now give a pvMO-1QFA $\mathcal{M}^l = (Q, \{a, b\}, \{U_a | \sigma \in \Sigma^l\}, \{0\}, Q_a, Q_f)$ to recognize $A^l$ exactly, where
• \( Q = \{ |0\rangle, |1\rangle, |2\rangle \}, Q_a = \{ |0\rangle \}, Q_r = \{ |1\rangle, |2\rangle \}. \\

• \( U_\sigma \) are defined as follows:

\[
U_q = \begin{pmatrix}
\alpha & -\beta & 0 \\
\beta & \alpha & 0 \\
0 & 0 & 1
\end{pmatrix},
U_a = \begin{pmatrix}
1 & 0 & 0 \\
0 & \cos \theta & \sin \theta \\
0 & -\sin \theta & \cos \theta
\end{pmatrix},
U_b = \begin{pmatrix}
1 & 0 & 0 \\
0 & \cos \theta & -\sin \theta \\
0 & \sin \theta & \cos \theta
\end{pmatrix},
U_\$ = U_k^{-1}.
\]

(17)

See [20] for more intuitions why we choose \( U_q \) and \( U_\$ \) in the way as above. Since \( U_a U_b = U_b U_a = I \), for \( w = \sigma_1 \ldots \sigma_{|w|} \in \{ a, b \}^* \), we have

\[
U_w = U_{\sigma_{|w|}} \ldots U_{\sigma_1} = U^{\#_a(w)}_a U^{\#_b(w)}_b.
\]

(18)

Let \( \#_a(w) = n \) and \( \#_b(w) = m \). If \( w \in A'_{yes} \), then the quantum state before the measurement is

\[
|q\rangle = U_\$ U_w U_q |0\rangle = U_\$ (U_a)^n (U_b)^m U_k |0\rangle = U_\$ (U_a)^n (U_b)^n U_k |0\rangle = |0\rangle.
\]

(19)

If the input \( w \in A'_{no} \) then the quantum state before the measurement is

\[
|q\rangle = U_\$ U_w U_q |0\rangle = U_\$ (U_a)^n (U_b)^m U_k |0\rangle = U_\$ (U_a)^n (U_b)^{n+1} U_k |0\rangle = U_\$ (U_b)^n U_k |0\rangle = \gamma_1 |1\rangle + \gamma_2 |2\rangle,
\]

(20)

where \( \gamma_1 \) and \( \gamma_2 \) are amplitudes that we do not need to specify more exactly.

Since the amplitude of \( |0\rangle \) in the above quantum state \( |q\rangle \) is 0, we get the exact result after the measurement of \( \gamma_1 |1\rangle + \gamma_2 |2\rangle \) in the standard basis \( \{ |0\rangle, |1\rangle, |2\rangle \} \). Therefore, we have

1. if \( w \in A'_{yes} \), then \( Pr[\mathcal{M}^\prime] \) accepts \( w \) = 1;
2. if \( w \in A'_{no} \), then \( Pr[\mathcal{M}^\prime] \) rejects \( w \) = 1.

We now prove the other direction. Namely, \( Pr[\mathcal{M}^\prime] \) accepts \( w \) = 1 implies that \( w \in A'_{yes} \).

Assume that \( w \notin A'_{yes} \), that is \( \#_a(w) \neq \#_b(w) \). The quantum state before the measurement is

\[
|q\rangle = U_\$ U_w U_q |0\rangle = U_\$ (U_a)^n (U_b)^m U_k |0\rangle = U_\$ (U_a)^{n-m} U_k |0\rangle
\]

(21)

\[
= \begin{pmatrix}
\alpha & \beta & 0 \\
-\beta & \alpha & 0 \\
0 & 0 & 1
\end{pmatrix} \begin{pmatrix}
1 & 0 & 0 \\
0 & \cos(m-n)\theta & -\sin(m-n)\theta \\
0 & -\sin(m-n)\theta & \cos(m-n)\theta
\end{pmatrix} \begin{pmatrix}
\alpha & -\beta & 0 \\
\beta & \alpha & 0 \\
0 & 0 & 1
\end{pmatrix} |1\rangle,
\]

(22)

\[
= \begin{pmatrix}
\alpha^2 + \beta^2 \cos(m-n)\theta \\
-\alpha \beta + \alpha \beta \cos(m-n)\theta \\
\beta \sin(m-n)\theta
\end{pmatrix}. \]

(23)

Since \( \theta = \sqrt{2}\pi \), there are no integers \( m \neq n \) such that \( \cos(m-n)\theta = 1 \). Therefore \( \alpha^2 + \beta^2 \cos(m-n)\theta \neq 1 \) and \( Pr[\mathcal{M}'] \) accepts \( w \neq 1 \).

We now prove the following: If \( Pr[\mathcal{M}'] \) rejects \( w \) = 1, then the input \( w \in A'_{no} \).

Assume that \( w \notin A'_{no} \), that is \( \#_a(w) \neq \#_b(w) + 1 \). The quantum state before the measurement is

\[
|q\rangle = U_\$ U_w U_q |0\rangle = U_\$ (U_a)^n (U_b)^m U_k |0\rangle = U_\$ (U_b)^{n-m} U_k |0\rangle = \begin{pmatrix}
\alpha^2 + \beta^2 \cos(m-n)\theta \\
-\alpha \beta + \alpha \beta \cos(m-n)\theta \\
\beta \sin(m-n)\theta
\end{pmatrix}.
\]

(24)
Let $m - n = l'$. Since $\theta = \sqrt{2}\pi$ and $m \neq n + l$, we have

$$
\alpha^2 + \beta^2 \cos(m - n)\theta = \alpha^2 + \beta^2 \cos l'\theta = \frac{-\cos l'\theta}{1 - \cos l'\theta} + \frac{1}{1 - \cos l'\theta} \cos l'\theta = \frac{\cos l'\theta - \cos l\theta}{1 - \cos l'\theta} \neq 0. \quad (25)
$$

Therefore, $Pr[M^l]$ accepts $w \neq 1$.

Hence, we have proved that the promise problem $A^l$ can be recognized exactly by the pvMO-1QFA $M^l$. Obviously, $A^l_{yes}$ and $A^l_{no}$ are not regular languages. According to Theorem 1, the promise problem $A^l$ cannot be recognized by any pvDFA.

\[ \square \]

Geffert and Yakaryılmaz \[16\] proved that the promise problem $\text{ExpEQ}(c)$ can be solved by a one-way probability finite automaton (PFA) $A(c)$, but there is no DFA solving $\text{ExpEQ}(c)$.

We define a new promise problem

$$
P\text{loyEQ} = \begin{cases} 
\text{PloyEQ}_{yes} = \{(a^n b^m \#)^t \mid n = m \text{ and } t \geq T\} & \\
\text{PloyEQ}_{no} = \{(a^n b^m \#)^t \mid n \neq m \text{ and } t \geq T\}
\end{cases}
$$

where $T$ is a polynomial of $l = \max\{n, m\}$ which will be given later.

**Theorem 13.** For any $\varepsilon \leq \frac{1}{3}$, the promise problem PloyEQ can be solved by a 1QCFA with the error probability $\varepsilon$, but there is no DFA solving the POLYPromiseEQ with the error probability $\varepsilon$.

**Proof.** Let $\theta = \sqrt{2}\pi$. We design a 1QCFA $M = (Q, S, \Theta, \Delta, \delta, |q_0\rangle, s_0, S_a, S_r)$ to solve the promise problem PloyEQ, where $Q = \{|0\rangle, |1\rangle\}$. The automaton proceeds as shown in Figure 1, where

$$
U_k = U_b = I, \quad U_a = \begin{pmatrix} \cos \theta & \sin \theta \\
-\sin \theta & \cos \theta \end{pmatrix}, \quad U_b = \begin{pmatrix} \cos \theta & -\sin \theta \\
\sin \theta & \cos \theta \end{pmatrix}. \quad (27)
$$

1. Read the left end-marker $\epsilon$, perform $U_k = I$ on the initial quantum state $|0\rangle$, do not change its classical state, and move the tape head one cell to the right.
2. Until the currently scanned symbol $\sigma$ is the right end-marker $\$$, do the following:
   2.1 If $\sigma \neq \#$, apply $\Theta(s_0, \sigma) = U_a$ to the current quantum state, do not change its classical state, and move the tape head one cell to the right.
   2.2 Otherwise, measure the current quantum state with $M = \{|0\rangle, |1\rangle\}$. If the outcome is $|1\rangle$, reject the input and halt. Otherwise, move the tape head one cell to the right.
3. Accept the input and halt.

Figure 1: The 1QCFA solving the promise problem POLYPromiseEQ.

Let us choose $T = \lceil 2T^2 \log_2 \frac{1}{\varepsilon} \rceil$. If the input $w \in \text{PloyEQ}_{yes}$, then the quantum state before measurement in the Step 2.2 is always $|0\rangle$. Therefore, the input will be accepted with certainty.

\[ \text{ExpEQ}(c) = (\text{ExpEQ}_{yes}(c), \text{ExpEQ}_{no}(c)) \text{ with } \text{ExpEQ}_{yes}(c) = \{(a^n b^m \#)^t \mid n = m \text{ and } c \rangle \mid m, n \in \mathbb{N}_+, m = n\} \text{ and } \text{ExpEQ}_{no}(c) = \{(a^n b^m \#)^t \mid m, n \in \mathbb{N}_+, m \neq n\}, \text{ where integer } c \geq 3. \]
If the input \( w \in \text{PloyEQ}_{\text{occ}} \), the quantum state before \( i \)-th measurement in the Step 2.2 is

\[
|q\rangle = U_a^n U_b^m = \left( \begin{array}{cc} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{array} \right)^n \left( \begin{array}{cc} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{array} \right)^m
\]

(28)

\[
= \left( \begin{array}{cc} \cos(m-n)\theta & -\sin(m-n)\theta \\ \sin(m-n)\theta & \cos(m-n)\theta \end{array} \right).
\]

(29)

According to [1, 36], the rejecting probability after the \( i \)-th measurement is therefore

\[
Pr_r > \frac{1}{2(m-n)^2} > \frac{1}{2l^2}.
\]

(30)

The probability that \( M \) rejects the input \( w \) is

\[
Pr[M \text{ rejects } w] = \sum_{i \geq 1} \left( Pr_r \prod_{i=1}^{i-1} (1 - Pr_r(i-1)) \right) > \sum_{i \geq 1} \left( \frac{1}{2l^2} \prod_{i=1}^{i-1} (1 - \frac{1}{2l^2}) \right)
\]

(31)

\[
= \sum_{i \geq 1} \frac{1}{2l^2} \left( 1 - \frac{1}{2l^2} \right)^{i-1} = \frac{1}{2l^2} \frac{1 - (1 - \frac{1}{2l^2})^i}{2l^2} = 1 - (1 - \frac{1}{2l^2})^i.
\]

(32)

Since \( 1 - x \leq e^{-x} \), we have

\[
Pr[M \text{ rejects } w] > 1 - (1 - \frac{1}{2l^2})^i > 1 - e^{-\frac{1}{2l^2}} \geq 1 - e^{-\frac{1}{2l^2} \log \frac{1}{\delta}} = 1 - e^{-\log \frac{1}{\delta}} = 1 - \epsilon.
\]

(33)

Therefore, the promise problem PloyEQ can be solved by a 1QCFA \( M \) with the error probability \( \epsilon \).

Assume now that there is a PFA \( A \) solving the POLYPromiseEQ with the error probability \( \epsilon \). Let us consider a 2PFA \( M \) running as follows:

1. \( M \) reads the input \( w \) from the left to the right;
2. After reading each \( \sigma \in \{a, b, \#\}, M \) simulates the transformation of the PFA \( A \) reading \( \sigma \);
3. When it reaches the right-end marker, \( M \) moves its tape head to the left most symbol of the input \( w \) and reads the input \( w \) again.

If \( M \) reads the input \( w \) \( T \) times, according to the assumption, then we have

\[
Pr[A \text{ accepts } a^n b^n \#] = Pr[A \text{ accepts } a^n b^n \#] \geq 1 - \epsilon
\]

(34)

and

\[
Pr[M \text{ accepts } a^n b^n \#] = Pr[A \text{ accepts } a^n b^n \#] \leq 1 - Pr[A \text{ rejects } a^n b^n \#] \leq \epsilon
\]

(35)

where \( n \neq m \).

Therefore, for any integers \( n \) and \( d > 0 \), it holds

\[
|Pr[M \text{ accepts } a^n b^n \#] - Pr[M \text{ accepts } a^n b^{n+d} \#]| \geq 1 - 2\epsilon \geq \epsilon.
\]

(36)

Since \( T \) is a polynomial of the length of the input \( w \), we can actually proved the following lemma (as in [12, 15]):

**Lemma 3.** Let \( \epsilon \leq \frac{1}{3} \). Suppose that \( M \) is a two-way probabilistic finite automaton (2PFA) with \( \exp(o(|w|)) \) expected running time, where \( |w| \) is the length of the input. Then there exists an integer \( d \), for all sufficiently large \( n \), such that

\[
|Pr[M \text{ accepts } a^n b^n \#] - Pr[M \text{ accepts } a^n b^{n+d} \#]| < \epsilon.
\]

(37)

Obviously, Equality (37) contradicts Equality (34). Therefore, there is no PFA solving POLYPromiseEQ with the error probability \( \epsilon \).
Acknowledgements

Some of S.G. Zheng’s work was done when visiting Sun Yat-sen University. S.G. Zheng would like to thank D.W. Qiu for his hospitality at Sun Yat-sen University. J. Gruska and S.G. Zheng were supported by the Employment of Newly Graduated Doctors of Science for Scientific Excellence project/grant (CZ.1.07./2.3.00 /30.0009) of Czech Republic. L.Z. Li was supported by the National Natural Science Foundation of China (Nos. 61100001, 61472452).

References

[1] A. Ambainis and J. Watrous, Two-way finite automata with quantum and classical states, Theoretical Computer Science 287 (2002) 299–311.
[2] A. Ambainis and R. Freivalds, One-way quantum finite automata: strengths, weaknesses and generalizations, in Proceedings of the 39th FOCS (1998) 332–341.
[3] A. Ambainis and N. Nahimovs, Improved constructions of quantum automata, Theoretical Computer Science 410 (2009) 1916–1922.
[4] A. Ambainis and A. Yakaryılmaz, Superiority of exact quantum automata for promise problems, Information Processing Letters 112 (2012) 289–291.
[5] A. Ambainis, Superlinear advantage for exact quantum algorithms, in Proceedings of 45th STOC (2013) 891–900.
[6] A. Ambainis, A. Iraids and J. Smotrovs, Exact quantum query complexity of EXACT and THRESHOLD, in Proceedings of 8th TQC (2013) 263–269. Also [arXiv:1302.1235]
[7] A. Ambainis, J. Gruska and S.G Zheng, Exact quantum algorithms have advantage for almost all Boolean functions, Quantum Information & Computation 15 (2015) 0435–0452 (to appear). Also [arXiv:1404.1684]
[8] A. Bertoni, C. Mereghetti and B. Palano, Golomb rulers and difference sets for succinct quantum automata, International Journal of Foundations of Computer Science 14 (2003) 871–888.
[9] A. Bertoni, C. Mereghetti and B. Palano, Small size quantum automata recognizing some regular languages, Theoretical Computer Science 340 (2005) 394–407.
[10] A. Bertoni, C. Mereghetti and B. Palano, Some formal tools for analyzing quantum automata, Theoretical Computer Science 356 (2006) 14–25.
[11] M.P. Bianchi, C. Mereghetti and B. Palano, Complexity of Promise Problems on Classical and Quantum Automata, Gruska Festschrift 2014, LNCS 8808, to apper.
[12] C. Dwork, L. Stockmeyer, Finite state verifiers I: The power of interaction, J. ACM 39 (4) (1992) 800–828.
[13] S. Even, A.L. Selman and Y. Yacobi, The Complexity of Promise Problems with Applications to Public-Key Cryptography, Information and Control 61 (1984) 159–173.
[14] R. Freivalds, M. Ozols and L. Mancinska, Improved constructions of mixed state quantum automata, Theoretical Computer Science 410 (2009) 1923–1931.
[15] A.G. Greenberg and A. Weiss, A lower bound for probabilistic algorithms for finite state machines, Journal of Computer and System Sciences 33 (1986) 88–103.
[16] V. Geffert, A. Yakaryılmaz, Classical automata on promise problems, In DCFS’14, LNCS 8614 (2014) 126–137. Also [arXiv:1405.6671]
[17] O. Goldreich, On promise problems: A survey, Shimon Even Festschrift, LNCS 3895 (2006) 254–290.
[18] J. Gruska, Quantum Computing, McGraw-Hill, London (1999).
[19] J. Gruska, D.W. Qiu and S.G. Zheng, Generalizations of the distributed Deutsch-Jozsa promise problem, [arXiv:1402.7254] (2014).
[20] J. Gruska, D.W. Qiu and S.G. Zheng, Potential of quantum finite automata with exact acceptance, Also [arXiv:1403.1689] (2014).
[21] J.E. Hopcroft and J.D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, New York (1979).
[22] M. Hirvensalo, Quantum automata with open time evolution, Int. J. Nat. Comput. Res. 1 (2010) 70–85.
[23] A. Kondacs and J. Watrous, On the power of quantum finite state automata, in Proceedings of the 38th FOCS (1997) 66–75.
[24] L.Z. Li, D.W. Qiu, X.F. Zou, L.J. Li, L.H. Wu and P. Mateus, Characterizations of one-way general quantum finite automata, Theoretical Computer Science 419 (2012) 73–91.
[25] A. Montanaro, R. Jozsa and G. Mitchison, On exact quantum query complexity, Algorithmica, DOI 10.1007/s00453-013-9826-8 (2013). Also arXiv:1111.0475.

[26] C. Moore and J.P. Crutchfield, Quantum automata and quantum grammars, Theoretical Computer Science 237 (2000) 275–306.

[27] M. Nakanishi, Quantum Pushdown Automata with a Garbage Tape, arXiv:1402.3449 (2014).

[28] M.A. Nielsen and I.L. Chuang, Quantum Computation and Quantum Information, Cambridge University Press, Cambridge (2000).

[29] J. Rashid and A. Yakaryılmaz, Implications of quantum automata for contextuality, In CIAA’14, LNCS 8587 (2014) 318–331. Also arXiv:1404.2761 (2014).

[30] P.W. Shor (1997), Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM Journal on Computing, 26, pp. 1484–1509. Earlier version in FOCS’94. Also arXiv:9508027.

[31] D. Simon , On the power of quantum computation, SIAM Journal on Computing, 26 (1997) 1474–1483. Earlier version in FOCS’94.

[32] J. Watrous, Quantum computational complexity, R.A. Meyers, Editor, Encyclopedia of Complexity and Systems Science, Springer, 2009, pp. 7174–7201.

[33] A. Yakaryılmaz and A.C. Cem Say, Succinctness of two-way probabilistic and quantum finite automata, Discrete Mathematics and Theoretical Computer Science 12 (2010) 19–40.

[34] S. Yu, Regular Languages, In: G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, Springer-Verlag, Berlin, 1998, pp. 41–110.

[35] S.G. Zheng, D.W. Qiu, L.Z. Li and J. Gruska, One-way finite automata with quantum and classical states, Dassow Festschrift 2012, LNCS 7300 (2012) 273–290.

[36] S.G. Zheng, D.W. Qiu, J. Gruska, L.Z. Li and P. Mateus, State succinctness of two-way finite automata with quantum and classical states, Theoretical Computer Science 499 (2013) 98–112.

[37] S.G. Zheng, J. Gruska and D.W. Qiu, Power of the interactive proof systems with verifiers modeled by semi-quantum two-way finite automata, arXiv:1304.3876 (2013).

[38] S.G. Zheng, J. Gruska and D.W. Qiu, On the state complexity of semi-quantum finite automata, RAIRO-Theoretical Informatics and Applications 48 (2014) 187–207. Earlier version in LATA’14.