UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Lucas Cavalcante de Sousa

Modelos Computacionais Quânticos

Florianópolis
2019

Lucas Cavalcante de Sousa

Modelos Computacionais Quânticos

Trabalho de Conclusão de Curso submetido ao Curso de Bacharelado em Ciências da Computação para a obtenção do Grau de Bacharel em Ciências da Computação.

Orientador: Prof. Dr. Eduardo Inacio Duzzioni

Coorientadora: Profª. Dra. Jerusa Marchi

Florianópolis
2019

RESUMO

A computação quântica vem evoluindo bastante nos últimos anos. O presente trabalho estuda a computação quântica por meio da utilização de máquinas abstratas que utilizam efeitos quânticos: as versões quânticas de autômatos finitos e de pilha. Esse trabalho apresenta alguns dos modelos existentes e suas propriedades conhecidas. Apresenta-se também exemplos de linguagens tratáveis pelo MO-1QFA, um autômato finito quântico com menor poder de reconhecimento, que ainda assim, reconhece algumas linguagens que sua versão clásica não reconhece. Esse trabalho também apresenta um estudo de caso explorando os erros atrelados a execução de um autômato finito quântico em uma plataforma quântica real.

Palavras-chave:
Modelos computacionais, modelos de máquinas abstratas, computação quântica, modelos computacionais quânticos

Sumário

Capítulo 1

Introdução

1.1 Computação e seus avanços

Em 1965, *Gordon. E. Moore*, cofundador da Intel, conjecturou [40] sobre o aumento da quantidade de transistores em um circuito integrado (um chip) e, consequentemente, sobre o aumento da performance dos processadores. *Moore* previu que o número de transistores em um processador dobraria anualmente. 10 anos depois, em 1975 [41], ele revisou sua previsão, falando que o número de transistores em um processador dobraria a cada 2 anos.

Hennessy e *Patterson* [26] analisaram o desenvolvimento de performance dos processadores de 1978 à 2018 (fig. 1) . Por meio desta análise percebemos que o crescimento descrito por *Moore* não ocorreu de fato, e mesmo com os vários avanços tecnológicos obtidos, ele não se manteve constante.

Entre 1978 e 1986, o crescimento anual observado foi de 25% - dobrando a cada 3,5 anos. Crescimento este atribuído ao barateamento do hardware e avanços de software, como: o avanço do microprocessador combinado com a redução do seu preço quando produzido em massa, o uso mais difundido de linguagens de programação (evitando programação em *Assembly* e reduzindo a necessidade de códigos direcionados à máquinas específicas,

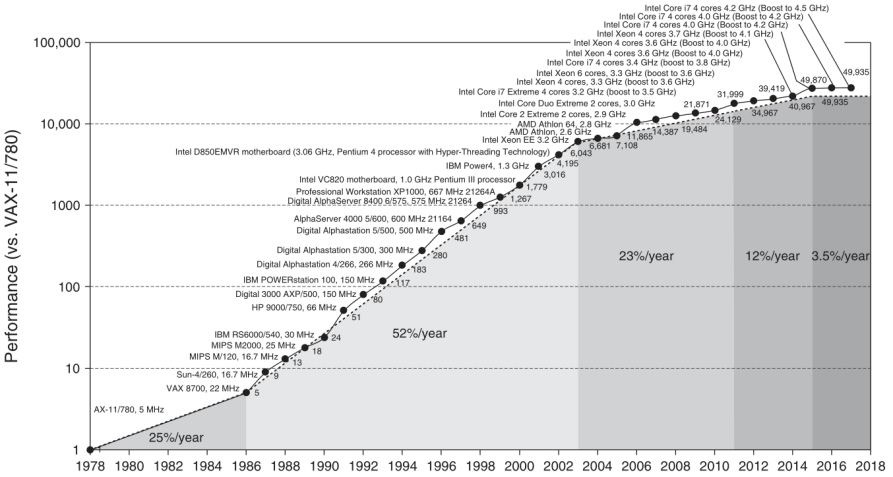

Figura 1 – Histórico de performance dos processadores de 1978 à 2018 [26].

facilitando o desenvolvimento e simplificando as camadas de abstração necessárias durante a programação), a criação de sistemas operacionais não proprietários (o que reduziu preço e o risco de produção arquitetural).

De 1986 à 2003, o crescimento anual de 52% - dobrando a cada 2 anos - foi atribuído aos avanços nas técnicas arquiteturais e organizacionais apresentadas pelos processadores RISC(*Reduced Instruction Set Computer*) e ao desafio que foi para as outras arquiteturas se equipararem à sua performance.

Até 2003, durante quase 17 anos, a conjectura de *Moore* se mostrou próxima da realidade. Graças a desenvolvimentos já mencionados, foi observado um crescimento médio anual de 50%, ou seja, dobrando a cada 2 anos. Porém, por volta de 2003, a lei da escalabilidade de *Dennard* [17] mostrou se aproximar do fim. Dennard observou, em 1974, uma correlação entre densidade energética, tamanho do chip e quantidade de transistores. Segundo *Dennard*, a densidade energética de um circuito se mantém constante mesmo com o aumento da quantidade de transistores graças à di-

minuição dos mesmos. Desta forma, contanto que os transistores usassem menos energia, eles poderiam continuar aumentando sua velocidade. Por volta de 2003, se mostrou inviável continuar diminuindo a densidade energética dos circuitos e manter a exatidão do sistema. Conforme se diminui a escala dos transistores, mais próximo se chega dos limites fundamentais físicos, o que causa interferências nas operações devido aos efeitos quânticos do tunelamento [44]. Desta forma um "0" pode virar um "1" e vice versa, mesmo que isso não seja desejado, causando tanto perda de informação quanto erros durante o processamento.

Desta forma, para conter essa diminuição do crescimento, foram difundidos os processadores multicores. Ainda assim, a partir de 2003 não foi observado o crescimento seguindo a lei de *Moore*: de 2003 à 2011 o crescimento foi de 23% ao ano - dobrando a cada 3.5 anos;

Isso se deve pela aplicação direta da lei de *Amdahl*, observada por *Amdahl* [12] em 1967, que expõe que uma aplicação nunca é 100% paralelizável, além disso, ela também expõe que o crescimento de performance de uma aplicação pelo acréscimo da quantidade de unidades de processamento está sujeito a um limite ligado ao quanto essa aplicação é paralelizável.

Assim de, 2011 à 2015, o crescimento foi de 12% ao ano - dobrando a cada 8 anos. E chegando no pior até o momento, onde de 2015 à 2018 o crescimento observado foi de 3,5% ao ano - dobrando a cada 20 anos.

Vendo a aproximação do fim da lei de *Moore* e da lei de escalabilidade de *Dennard*, aliado ao limite imposto pela lei de *Amdahl*, surge a necessidade de um novo impulso à computação. Várias formas foram propostas, como biocomputação [22] , o uso de grafeno para fabricação dos circuitos [37] e a computação quântica [21]. O presente trabalho tratará sobre a computação quântica [44].

1.2 O porquê da Computação Quântica

A computação quântica reúne conceitos físicos da mecânica quântica, como superposição e entrelaçamento, à computação clássica, como nos referenciaremos à computação atual. Graças a esses efeitos físicos, a computação quântica se apresenta como uma forma de expandir a barreira do que é tratável atualmente [44].

A ideia de um computador quântico foi inicialmente proposta em 1982 por *Feynman* [21], após ele propor a simulação de sistemas físicos quânticos em um computador universal - tarefa que se mostra difícil mesmo nas máquinas atuais, devido ao crescimento exponencial requerido para tal simulação. Após *Feynman*, vieram outros avanços importantes na computação quântica que demonstraram prospectos para a área: em 1994, *Shor* [57] desenvolveu um algoritmo quântico de tempo polinomial para fatoração e, portanto, tratável em um computador quântico, o que trouxe grande atenção para a área, pelo risco que traz aos sistemas de segurança atuais, como o algoritmo criptográfico RSA [53] [57] e criptografia de curvas elípticas [38] [54]; em 1996, *Grover* [24] desenvolveu um algoritmo de busca em base de dados não estruturada com ganho quadrático do tamanho da estrutura com relação a operação clássica.

Em 1985, *Deutsh* [18] descreveu pela primeira vez a junção da mecânica quântica com um modelo computacional estudado na teoria da computação clássica. *Deutsh* propôs uma versão quântica da Máquina de Turing e, baseado na tese de *Church-Turing*, propôs também o princípio de *ChurchTuringDeutsch* que declara que com esse modelo universal seria possível simular qualquer processo físico, como inicialmente idealizado por *Feynman*. A partir disso, foram criados outros modelos quânticos das máquinas estudadas em teoria da computação.

1.3 Objetivos

Tem-se como objetivo geral o estudo de versões quânticas dos autômatos, e a sua experimentação em um computador quântico real.

Objetivo Específico

O presente trabalho tem como objetivos específicos:

- realizar revisão sobre os modelos existentes e as relações conhecidas entre eles e as versões clássicas;
- Exemplificar problemas tratáveis com um dos modelos de autômato finito quântico, o MO-1QFA;
- Análise de erro do MO-1QFA em uma plataforma do IBMQ [29].

1.4 Estrutura do Trabalho

Este trabalho primeiramente propõe uma revisão teórica dos temas pertinentes ao estudo dos modelos computacionais quânticos, iniciando por uma introdução aos modelos clássicos (autômatos finitos (seção 2.1.1) e autômatos de pilha (seção 2.1.2)), à álgebra linear (seção 2.2) e à mecânica quântica (seção 2.3), em seguida, é apresentado um histórico dos estudos dos autômatos quânticos conhecidos(seção 3.2), apresentação da definição e de problemas tratáveis pelo MO-1QFA (seção 3.2) e por fim uma experimentação de um dos autômatos apresentados em uma plataforma quântica real(seção 4.1).

Capítulo 2

Revisão Teórica

Neste capítulo se concentra toda a revisão teórica referente aos temas pertinentes ao trabalho, iniciando por uma introdução aos autômatos finitos (seção 2.1.1) e autômatos de pilha (seção 2.1.2)), à álgebra linear (seção 2.2) e à mecânica quântica (seção 2.3).

2.1 Modelos Computacionais Clássicos

Antes do estudo das versões quânticas dos modelos, é importante possuir um embasamento sobre os modelos clássicos. A referência deste capítulo é *Introduction to the Theory of Computation* [58] e *Introduction to automata theory, languages, and computation* [28], aqui são abordados as definições, representações e alguns exemplos de autômatos finitos (seção 2.1.1) e autômatos de pilha(seção 2.1.2).

2.1.1 Autômatos Finitos

Em geral, um autômato finito (AF) é representado por um diagrama de estados, por exemplo, um AF que aceita a linguagem L, que é o conjunto das palavras constituídas somente por símbolos a e b onde a quantidade de a's é par e não existem 2 a's consecutivos pode ser representado como na fig. 2.

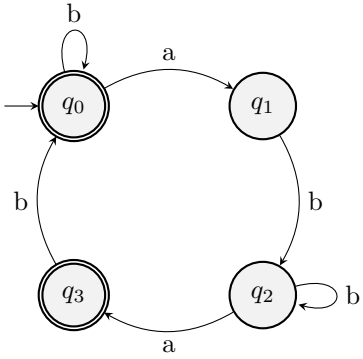

Figura 2 – Autômato M_0 que reconhece uma quantidade par de a's, onde não existem 2 a's consecutivos.

A estrutura básica de um autômato é constituída de estados e de transições. Os estados são as circunferências (q_0, q_1, q_2, q_3), que indicam pontos da computação. Além disso, os estados podem ser ainda mais específicos, sendo um estado inicial ou final. O estado inicial, representado pela seta (q_0), é onde se inicia a computação do autômato, o estado que o autômato se encontra no tempo 0. Os estados finais, representado pelas circunferências concêntricas (q_0, q_3), são estados que indicam se a entrada será aceita ou não ao final da computação. Além dos estados, um autômato também possui transições, representadas por setas ligando estados. As transições mostram que a partir de um determinado estado, com a leitura de um determinado símbolo o estado será alterado para o apontado pela seta.

Durante seu funcionamento, um autômato inicia no estado inicial, e lê a palavra de entrada, símbolo a símbolo da esquerda para direita, e a cada símbolo segue as transições do estado atual para alterar de estado. Se ao fim da palavra, o estado for final, a entrada é dita aceita e, portanto reconhecida pelo autômato. Caso ele não seja final, a palavra é dita rejeitada e, portanto não reconhecida pelo autômato.

Por exemplo, ao receber *aba* como entrada, M_0 passa pelo seguinte processamento:

1. Inicia no estado q_0;
2. Lê a e transita de q_0 para q_1;
3. Lê b e transita de q_1 para q_2;
4. Lê a e transita de q_2 para q_3;
5. Aceita, já que ao final do processamento da entrada M_0 se encontra no estado q_3, que é final.

E ao receber a palavra *ababba* como entrada, M_0 passa pelo seguinte processamento:

1. Inicia no estado q_0;
2. Lê a e transita de q_0 para q_1;
3. Lê b e transita de q_1 para q_2;
4. Lê a e transita de q_2 para q_3;
5. Lê b e transita de q_3 para q_0;
6. Lê b e transita de q_0 para q_0;
7. Lê a e transita de q_0 para q_1;
8. Rejeita, já que ao final do processamento da entrada M_0 se encontra no estado q_1, que não é final.

Definição Formal

Definição 1. *Um autômato finito é dado pela quíntupla* $M = (Q, \Sigma, \delta, q_0, F)$*:*

Q *é o conjunto finito dos estados;*

Σ *é o conjunto finito do alfabeto de entrada;*

$\delta : Q \times \Sigma \to Q$ *é a função transição;*

$q_0 \in Q$ *é o estado inicial do autômato;*

$F \subseteq Q$ *é o conjunto de estados finais.*

A função de transição δ define um mapeamento de cada estado, sobre cada símbolo, para um estado. Para evitar a poluição dos autômatos, em geral se omitem as transições irrelevantes. Por exemplo, na fig. 2, podemos ver que para q_1 não existe nenhuma transição definida pelo símbolo a, isto é, se o estado atual é q_1, ao ler um a da entrada se deve rejeitar a entrada. Em alguns momentos pode ser interessante a representação completo, ou seja, o autômato com todas as transições.

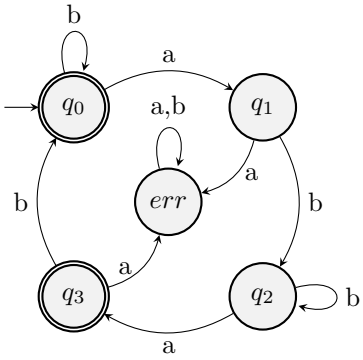

Figura 3 – M_1, versão completa de M_0.

Na fig. 3 podemos ver que M_1 é um autômato completo: todos os estados possuem transições definidas com todos os símbolos de entrada, e que para continuar com o mesmo propósito, foi criado um estado *err* de erro, para o qual todas as transições que representam uma má formação com relação a regra de formação das palavras aceitas vão.

Outras formas de representação

Existem outras formas de se representar um autômato, este trabalho apresenta a representação por meio de tabela e usará a por meio da notação

de *Dirac*, que é uma representação mais próxima das usadas nos autômatos finitos quânticos.

Na tabela 1, podemos ver um AF com sua função de transição em forma de tabela. Partindo de qualquer estado (primeira coluna), para qual estado M_0 transita ao receber cada símbolo de entrada (segunda e terceira coluna). Podemos ver também o estado inicial, que possui uma $\rightarrow$ ao seu lado e, os estados finais, que possuem um $*$.

δ	a	b
$\rightarrow q_0$	q_1	q_0
q_1	-	q_2
q_2	q_3	q_2
q_3	-	q_0

Tabela 1 – Função transição de M_0.

Neste trabalho utilizaremos a notação de *Dirac* como forma de representação dos autômatos finitos. Utilizando a notação de *Dirac* podemos ignorar os efeitos da mecânica quântica e aproximar os AF's visto até agora para suas versões quânticas. Nessa representação os estados serão vetores:

$$q_0 = |0\rangle \, ; q_1 = |1\rangle \, ; q_2 = |2\rangle \, ; q_3 = |3\rangle \, , \tag{2.1}$$

a função transição será dada através de um conjunto de matrizes, uma para cada símbolo de entrada:

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} ; B = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} , \tag{2.2}$$

onde a matriz A(resp. B) da eq. (2.2) representa todas as transições que possuem o símbolo a (resp. b) como gatilho. A posição A(i,j) (resp. B(i,j)), i-ésima linha e j-ésima coluna, de A (resp. B) tem o valor 1 quando o estado j transita para o estado i por a (resp. b), e possui o valor 0 caso contrário.

A transição de estados é dada pela multiplicação do estado atual pela matriz de transição do símbolo de entrada sendo lido, desta forma, dada que o autômato se encontra no estado $|3\rangle$ e lê um b ele transita para o seguinte estado:

$$B\,|3\rangle = |0\rangle = q_0. \tag{2.3}$$

Dada uma computação intermediária, de *ababba* partindo do estado $|2\rangle$, o estado final da computação será:

$$ABBABA\,|2\rangle = |3\rangle = q_3, \tag{2.4}$$

de forma similar, podemos definir operadores de projeção (seção 2.2.4) dos estados finais para avaliar se a palavra é aceita:

$$P_{\text{acc}} = \sum_{i \in F} |i\rangle\langle i|\,. \tag{2.5}$$

Agora podemos definir a computação como a multiplicação do operador de projeção dos estados finais pela computação apresentada anteriormente, caso o resultado seja 1, a palavra é aceita, caso o resultado a palavra seja 0, ela é rejeitada. Desta forma a computação de *ababba* é dada por:

$$P_{\text{acc}} ABBABA\,|0\rangle = 0, \tag{2.6}$$

e portanto, *ababba* não é aceita por M_0. Podemos observar na eq. (2.6) que as matrizes estão na ordem inversa da palavra, isso se deve pela ordem da aplicação das matrizes, uma matriz afeta o estado a sua direita. Portanto, quanto mais próximo do início da palavra está uma letra mais a direita na equação da computação estará sua matriz de transição.

O autômato finito visto até o momento é chamado de autômato finito determinístico, pois para cada par de estado e símbolo de entrada existe uma transição, com a notação de *Dirac* isso se concretiza possibilitando somente um valor 1 por coluna nas matrizes de transição. Existem outras versões do autômato finito como o autômato finito não determinístico [58] e o autômato finito probabilístico [52]. Essas versões não são relevantes

para o presente trabalho, porém, são interessantes para demonstrar a simplicidade da notação de *Dirac* nos autômatos.

Os autômatos finitos não determinísticos podem definir múltiplas transições partindo de um mesmo estado pelo mesmo símbolo de entrada, o que permite ele a estar em múltiplos estados em um dado instante da computação. Podemos adicionar o não determinismo nas matrizes de transição simplesmente permitindo mais de um valor 1 em suas colunas. Ao aplicar o operador de projeção dos estados finais, caso o resultado seja maior ou igual a 1 a palavra é aceita.

Já os autômatos probabilísticos possuem uma probabilidade atrelada a cada transição, onde as soma desses valores atrelados às transições partindo de um mesmo estado e, por um mesmo símbolo somam 1. Podemos descrever autômatos probabilísticos ao permitir múltiplos valores em uma mesma coluna, onde a soma destes, sempre somam 1. Ao aplicar o operador de projeção dos estados finais, obtemos a probabilidade da palavra ser aceita.

Problema da paridade

O problema da paridade consiste em dizer se uma determinada parte da palavra ocorre uma quantidade par ou ímpar de vezes. Por simplicidade, vamos considerar a substring a em um alfabeto de a's e b's. Assim nosso autômato M pode ser definido por $M = (Q, \Sigma, \delta, q_0, F)$:

Q é $\{|0\rangle,|1\rangle\}$;

Σ é {a,b};

δ é dado pelas seguintes matrizes de dimensões 2×2:

$$\begin{aligned} A &= |0\rangle\langle 1| + |1\rangle\langle 0|\,; \\ B &= I; \end{aligned} \tag{2.7}$$

onde I é a matriz identidade;

q_0 é $|0\rangle$;

F é $\{|0\rangle\}$.

O autômato se encontra no estado $|0\rangle$(resp. $|1\rangle$) quando possui uma quantidade par(resp. ímpar) de a's. A matriz A faz esse papel de alternar entre os estados a cada a, e a matriz B deixa o estado inalterado. Se ao final da palavra houver uma quantidade par de a's o estado após a computação será $|0\rangle$, um estado final, portanto ao aplicar o operador de projeção dos estados finais o resultado será 1 e a palavra é aceita. Caso contrário, o resultado será 0, portanto, a palavra será rejeitada.

A computação de *ababab* é dada por:

$$P_{\text{acc}}BABABAq_0, \tag{2.8}$$

onde P_{acc} é dado por:

$$P_{\text{acc}} = \sum_{i\in F} |i\rangle\langle i| = |0\rangle\langle 0|\,, \tag{2.9}$$

e computação seria:

$$\begin{aligned} P_{\text{acc}}BABABAq_0 &= P_{\text{acc}}BABABA\,|0\rangle \\ &= P_{\text{acc}}BABAB\,|1\rangle = P_{\text{acc}}BABA\,|1\rangle \\ &= P_{\text{acc}}BAB\,|0\rangle = P_{\text{acc}}BA\,|0\rangle \\ &= P_{\text{acc}}B\,|1\rangle = |0\rangle\langle 0|\,|1\rangle = 0; \end{aligned} \tag{2.10}$$

como o resultado é 0, *ababab* não é aceita por M.

Vamos agora considerar *bbbabaaa* e sua computação:

$$\begin{aligned} P_{\text{acc}}AAABABBBq_0 &= P_{\text{acc}}AAABABBB\,|0\rangle \\ &= P_{\text{acc}}AAABABB\,|0\rangle = P_{\text{acc}}AAABAB\,|0\rangle \\ &= P_{\text{acc}}AAABA\,|0\rangle = P_{\text{acc}}AAAB\,|1\rangle \\ &= P_{\text{acc}}AAA\,|1\rangle = P_{\text{acc}}AA\,|0\rangle \\ &= P_{\text{acc}}A\,|1\rangle = |0\rangle\langle 0|\,|0\rangle = 1, \end{aligned} \tag{2.11}$$

como temos o resultado 1, *bbbabaaa* é aceita por M.

É interessante notar que este AF é facilmente convertido para aceitar uma quantidade ímpar de a's, bastando somente alterar o conjunto de estados finais para $F = \{|1\rangle\}$ ao invés de $F = \{|0\rangle\}$.

Problema do módulo

O problema do módulo m consiste em verificar se uma determinada parte da palavra ocorre um número múltiplo de m vezes. Novamente para simplificar os exemplos, usaremos o alfabeto $\{a, b\}$ e a como a substring desejada. Desta forma, podemos definir $\text{MOD}^m = (Q, \Sigma, \delta, q_0, F)$ de maneira genérica:

Q é $\{|0\rangle, |1\rangle, \ldots, |m-1\rangle\}$;

Σ é {a,b};

δ é dado pelas seguintes matrizes de dimensões $m \times m$

$$A = \sum_{i=1}^{m} |i\%m\rangle\langle i-1)| \tag{2.12}$$

, onde "%" é a operação módulo;

$$B = I; \tag{2.13}$$

, onde I é a matriz identidade;

q_0 é $|0\rangle$;

F é $\{|0\rangle\}$.

A matriz B nos garante que não importa quantos b's a palavra tenha o estado continuará inalterado. Já a matriz A nos garante a contagem de a's ao alternar entre os estados de forma cíclica, ou seja, alternando entre $|0\rangle, |1\rangle, \ldots, |m-2\rangle, |m-1\rangle, |0\rangle$ e assim por diante até o fim da entrada. Assim o estado do autômato será $|i\rangle$ quando a quantidade de a's módulo m lidos for igual a i. E a palavra será aceita se o estado final for $|0\rangle$, ou

seja, quando a quantidade de a's módulo m for igual a 0, caso contrário a palavra é rejeitada.

Como exemplo, vamos considerar $\mathrm{MOD}^4 = (Q, \Sigma, \delta, q_0, F)$:

$$A = \sum_{i=1}^{m} |i\%m\rangle\langle i-1| = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}; \qquad B = I; \tag{2.14}$$

A computação de *aaaaab* é dada por:

$$P_{\mathrm{acc}} AAAAAB q_0, \tag{2.15}$$

onde P_{acc} é dado por:

$$P_{\mathrm{acc}} = \sum_{i \in F} |i\rangle\langle i| = |0\rangle\langle 0|; \tag{2.16}$$

e computação é:

$$\begin{aligned} P_{\mathrm{acc}} AAAAAB q_0 &= P_{\mathrm{acc}} A^5 B |0\rangle \\ &= P_{\mathrm{acc}} A^5 |0\rangle = P_{\mathrm{acc}} A^4 |1\rangle \\ &= P_{\mathrm{acc}} A^3 |2\rangle = P_{\mathrm{acc}} A^2 |3\rangle \\ &= P_{\mathrm{acc}} A |0\rangle = |0\rangle\langle 0| |1\rangle = 0, \end{aligned} \tag{2.17}$$

como o resultado é 0, *aaaaab* não é aceita por M.

Vamos agora considerar *bbbababaa* a computação seria:

$$\begin{aligned} P_{\mathrm{acc}} AABABABBB q_0 &= P_{\mathrm{acc}} AABABABBB |0\rangle \\ &= P_{\mathrm{acc}} AABABABB |0\rangle = P_{\mathrm{acc}} AABABAB |0\rangle \\ &= P_{\mathrm{acc}} AABABA |0\rangle = P_{\mathrm{acc}} AABAB |1\rangle \\ &= P_{\mathrm{acc}} AABA |1\rangle = P_{\mathrm{acc}} AAB |2\rangle \\ &= P_{\mathrm{acc}} AA |2\rangle = P_{\mathrm{acc}} A |3\rangle \\ &= |0\rangle\langle 0| |0\rangle = 1, \end{aligned} \tag{2.18}$$

como temos o resultado 1, *bbbababaa* é aceita por M.

Feita a revisão dos autômatos, seu funcionamento e a apresentação da notação de *Dirac* para aproximação dos AF's para os modelos quânticos, será feita uma revisão sobre a álgebra linear necessária no estudo da computação quântica.

2.1.2 Autômatos de Pilha

A grande vantagem dos autômatos de pilha (AP) quando comparados com os AF's é que além dos estados, os autômatos de pilha possuem uma memória em formato de pilha, é possível armazenar infinitas informações, porém só é possível acessar a última informação armazenada.

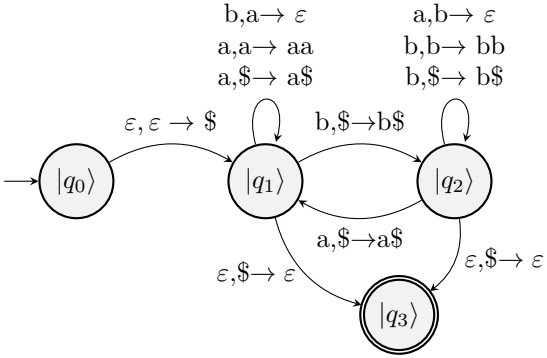

Figura 4 – Autômato que reconhece palavras com a mesma quantidade de a's e b's.

A diferença de um AF e de um AP é a função transição. Em fig. 4, as transições além de terem a informação do símbolo de entrada, possuem também o estado da memória como parte do gatilho.

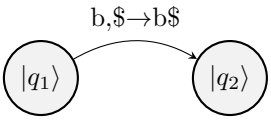

Figura 5 – Transição de $|q_1\rangle$ para $|q_2\rangle$ do autômato da fig. 4.

Tomando por exemplo a transição de $|q_1\rangle$ para $|q_2\rangle$ na fig. 5. O b antes da vírgula é símbolo lido da palavra de entrada; após a vírgula temos as informações com relação a manutenção da pilha: antes da seta temos o símbolo que esperamos que esteja na pilha para fazer a transição, ou seja, agora o gatilho não depende só do estado e símbolo da entrada, mas também do símbolo lido da pilha; após a seta temos a alteração que ocorrerá na pilha. Portanto esta transição específica que estando no estado $|q_1\rangle$, lendo b da palavra de entrada e lendo '$' da pilha, o estado do AP é alterado para $|q_2\rangle$ e um b é adicionado ao topo da pilha.

O símbolo ε representa a palavra vazia, que é contido infinitas vezes entre quaisquer 2 caracteres. Quando antes da vírgula, representa que a entrada não será lida para efetuar esta transição. Quando na seção de manutenção da pilha, pode significar duas coisas, que o símbolo antes da seta será desempilhado, como em $a, b \rightarrow \varepsilon$ ou que a pilha não será lida, como em $\varepsilon, \varepsilon \rightarrow \$$. Entre as várias versões, este modelo termina sua execução ao atingir um estado final, ler todos os caracteres da entrada e não possuir nada na pilha. Para tal, utilizamos o símbolo $\$|\$ \notin \Sigma$ como o fundo de pilha. Sempre que é possível ler $ sabemos que este é o único caractere na pilha.

Quando o AP da fig. 4 recebe uma entrada ele processa esse valor e, como um AF, tem como saída aceita ou rejeita. Ele inicia seu processamento em $|q_0\rangle$, seu estado inicial, computando um símbolo de cada vez até o final da entrada. Para cada símbolo, o próximo estado é obtido com base no símbolo lido da entrada, no símbolo lido do topo da pilha e no estado atual. Ao processar toda a palavra, ela é dita aceita por M_0 se M_0 estiver num estado final com a pilha vazia, e é dita rejeitada por M_0 caso contrário.

Computação

Ao receber *ababba* como entrada o processamento é o seguinte:

1. Inicia no estado $|q_0\rangle$;
2. Não lê a entrada, escreve '$' na pilha e transita de $|q_0\rangle$ para $|q_1\rangle$;
3. Lê a na entrada, lê um $ na pilha, portanto, empilha um a e transita de $|q_1\rangle$ para $|q_1\rangle$;
4. Lê b na entrada, lê um a na pilha, portanto, desempilha um a e transita de $|q_1\rangle$ para $|q_1\rangle$;
5. Lê a na entrada, lê um $ na pilha, portanto, empilha um a e transita de $|q_1\rangle$ para $|q_1\rangle$;
6. Lê b na entrada, lê um a na pilha, portanto, desempilha um a e transita de $|q_1\rangle$ para $|q_1\rangle$;
7. Lê b na entrada, lê um $ na pilha, portanto, empilha um b e transita de $|q_1\rangle$ para $|q_2\rangle$;
8. Lê a na entrada, lê um b na pilha, portanto, desempilha um b e transita de $|q_2\rangle$ para $|q_2\rangle$;
9. Não lê a entrada, lê um $ na pilha, portanto desempilha um $ e transita de $|q_2\rangle$ para $|q_3\rangle$;
10. Não há mais entrada, a pilha está vazia, e o estado atual é final, portanto *ababba* é aceita.

Uma forma um pouco mais completa de representar está computação seria através de uma árvore, como exemplo, a computação de *aba* (fig. 6).

Aqui podemos ver os momentos em que, não deterministicamente, a máquina toma mais de uma decisão. É importante notar que, para um AP aceitar uma palavra pelo menos um dos ramos da sua árvore de computação precisa aceitar a palavra. E que para rejeitar, todos os ramos precisam rejeitar. Também é possível notar que com o não determinismo, parte da computação ocorre em paralelo, o que faz com que o tempo de computação seja linear com relação ao tamanho da palavra.

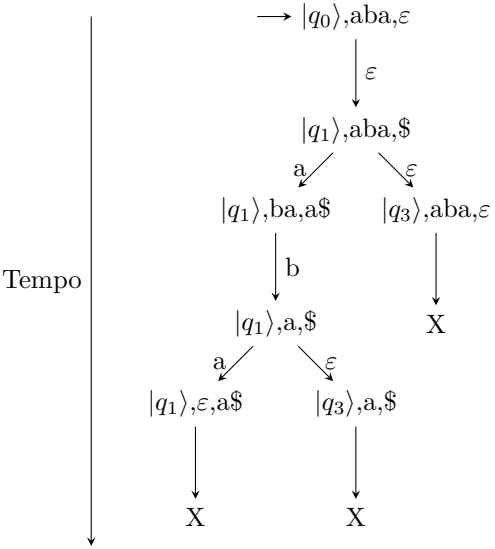

Figura 6 – Autômato M_0 reconhece palavras com a mesma quantidade de a's e bs.

Na árvore cada nodo tem as informações do estado atual, a entrada restante e a pilha. As setas entre os nodos indicam o que foi lido da entrada. O X indica que aquele ramo da computação gerou erro.

Definição Formal

Definição 2. *Um autômato de pilha é dado pela sêxtupla* $M = (Q, \Sigma, \Gamma, \delta, |q_0\rangle, F)$*:*

Q *é o conjunto finito dos estados;*

Σ *é o conjunto finito do alfabeto de entrada;*

Γ *é o conjunto finito do alfabeto de pilha;*

$\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to P(Q \times \Gamma_\varepsilon)$ *é a função transição;*

$|q_0\rangle \in Q$ *é o estado inicial do autômato;*

$F \subseteq Q$ *é o conjunto de estados finais.*

Aqui é necessário que haja transições por ε para obter todo o poder computacional de um AP (já que um AP determinístico reconhece um subconjunto do que é reconhecido por AP não determinístico [58]) e portanto temos: $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$, e como dito anteriormente, ε é utilizado na manutenção da pilha, e para isso temos $\Gamma_\varepsilon = \Gamma \cup \{\varepsilon\}$.

A função transição agora tem como gatilho o estado atual, a leitura da entrada(adicionado ε) e a leitura da pilha(adicionado ε) e possui como saída um conjunto de estados e ações de manutenção na pilha.

Número de a's maior que número de b's

Com poucas alterações no AP da fig. 4 podemos obter um autômato para reconhecer quando o número de a's é maior que o número de b's:

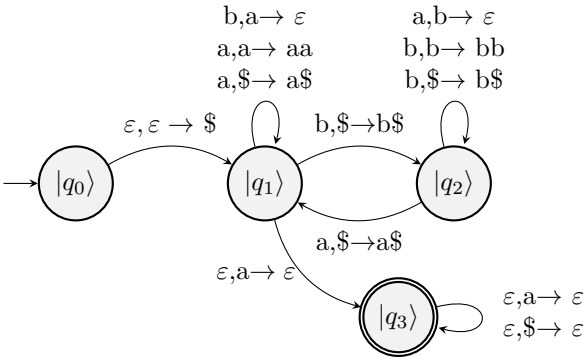

Figura 7 – Autômato que reconhece palavras com maior quantidade de a's que 'b's.

No AP da fig. 7 a contagem de a's é empilhada e para ir ao estado final é necessário que haja algum a na pilha, ou seja, até o momento foram vistos mais a's que b's, em seguida a pilha é esvaziada.

De maneira equivalente é possível construir um AP para reconhecer um número maior de b's do que a's (fig. 8).

Podemos também construir um AP para reconhecer um número diferente de a's e b's (fig. 9).

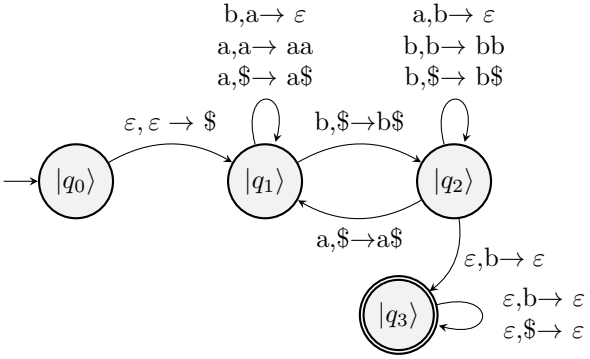

Figura 8 – Autômato que reconhece palavras com quantidade de a's menor que de b's.

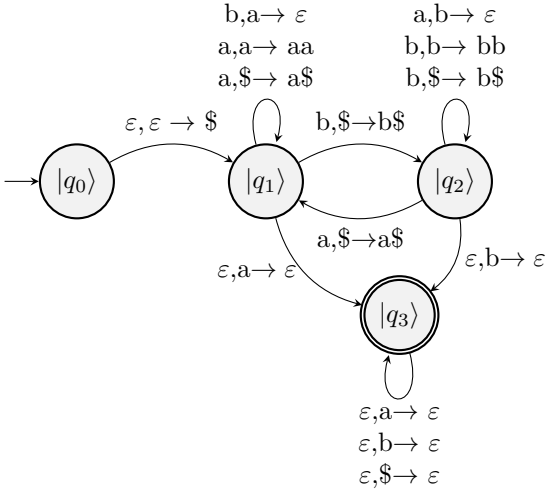

Figura 9 – Autômato que reconhece palavras com quantidade diferente de a's e b's.

2.2 Álgebra Linear para Computação Quântica

A álgebra linear tem suma importância na computação quântica. Dessa forma, nesta seção será feita uma breve revisão da álgebra linear e algumas propriedades utilizadas nos estudos em computação quântica e, consequentemente, no estudo dos modelos computacionais que serão apresentados.

Além disso, será apresentada a notação *Dirac*, a notação utilizada na mecânica quântica para descrever os estados quânticos.

Em geral o interesse da computação quântica é voltado a espaços vetoriais de dimensão finita sobre os números complexos($\mathbb{C}$), desta forma, a revisão apresentada aqui se vê sobre este corpo.

A referência deste capítulo é a segunda seção de *Nielsen* e *Chuang* [44], onde é abordada a álgebra linear para computação quântica. Aqui alguns desses tópicos são revisados, entre eles: espaço vetorial (seção 2.2.1), base e independência linear (seção 2.2.2), algumas operações relevantes referentes aos espaços vetoriais (seção 2.2.3), transformações lineares (seção 2.2.4), alguns operadores lineares relevantes (seção 2.2.5) e produto tensorial(seção 2.2.6).

2.2.1 Espaço Vetorial

Na notação de *Dirac*, também conhecida como notação *braket*, um estado quântico é um vetor coluna e é representado dentro de um *ket* $|\rangle$. Então dado um estado quântico ψ pertencente à $\mathbb{C}^n$ (vetores de dimensão n composto por elementos complexos) temos:

$$|\psi\rangle = (\psi_0 \ , \psi_1 \ , \ \dots \ , \psi_{n-1}) = \begin{bmatrix} \psi_0 \\ \psi_1 \\ \vdots \\ \psi_{n-1} \end{bmatrix}. \tag{2.19}$$

Um espaço vetorial V sobre $\mathbb{C}^n$ é um conjunto de vetores de dimensão n compostos por elementos complexos e equipado com as operações indicadas abaixo. As operações a seguir já demonstram parte da notação de *Dirac*.

Para $|\psi\rangle$, $|\zeta\rangle$, $|\varphi\rangle \in \mathbb{C}^n$ e α, $\beta \in \mathbb{C}$ temos:

soma entre vetores

$$|\psi\rangle + |\varphi\rangle = |\varphi\rangle + |\psi\rangle = \begin{bmatrix} \psi_0 + \varphi_0 \\ \psi_1 + \varphi_1 \\ \vdots \\ \psi_{n-1} + \varphi_{n-1} \end{bmatrix}; \tag{2.20}$$

$$(|\psi\rangle + |\varphi\rangle) + |\zeta\rangle = |\psi\rangle + (|\varphi\rangle + |\zeta\rangle); \tag{2.21}$$

$$|\psi\rangle + 0 = \begin{bmatrix} \psi_0 + 0 \\ \psi_1 + 0 \\ \vdots \\ \psi_{n-1} + 0 \end{bmatrix} = |\psi\rangle\,; \tag{2.22}$$

$$|\psi\rangle - |\psi\rangle = \begin{bmatrix} \psi_0 - \psi_0 \\ \psi_1 - \psi_1 \\ \vdots \\ \psi_{n-1} - \psi_{n-1} \end{bmatrix} = 0. \tag{2.23}$$

multiplicação de vetor por escalar

$$\alpha\,|\psi\rangle = \begin{bmatrix} \alpha\psi_0 \\ \alpha\psi_1 \\ \vdots \\ \alpha\psi_{n-1} \end{bmatrix}; \tag{2.24}$$

$$(\alpha\beta)\,|\psi\rangle = \alpha(\beta\,|\psi\rangle); \tag{2.25}$$

$$\alpha(|\psi\rangle + |\varphi\rangle) = \alpha\,|\psi\rangle + \alpha\,|\varphi\rangle\,; \tag{2.26}$$

$$(\alpha + \beta)\,|\psi\rangle = \alpha\,|\psi\rangle + \beta\,|\psi\rangle\,. \tag{2.27}$$

2.2.2 Base Computacional e Independência Linear

Um espaço vetorial pode ser gerado a partir da combinação linear de um conjunto de vetores, este conjunto é chamado de base do espaço.

Uma base muito utilizada neste trabalho é a base computacional. Os vetores da base computacional de dimensão n são:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, |1\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, |2\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \ldots, |n-2\rangle = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \end{bmatrix}, |n-1\rangle = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 1 \end{bmatrix}. \tag{2.28}$$

Todos os vetores pertencentes a um espaço vetorial podem ser gerados a partir de uma combinação linear dos vetores de sua base. Desta forma, dado um estado ψ que pertence a base computacional e $\alpha_0, \alpha_1, \ldots, \alpha_n \in \mathbb{C}$, $|\psi\rangle$ pode ser descrito como:

$$|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle + \ldots + \alpha_n |n\rangle . \tag{2.29}$$

Espaço de Hilbert

Na computação quântica, em geral, trabalhamos sobre espaços vetoriais chamados de espaço de Hilbert, que são espaços vetoriais munidos com a operação de produto interno (seção 2.2.3) e com a propriedade de completude. Como, ao trabalhar em sistemas quânticos, utilizamos espaços de Hilbert com dimensões finitas, a propriedade de completude é automática.

2.2.3 Operações relevantes

Nesta seção, é apresentado um conjunto de operações relevantes aos estudos de espaços vetoriais.

Produto Interno

Se tratando de espaços reais, o produto interno correlaciona o tamanho de dois vetores ao ângulo entre eles, retornando um escalar:

$$(|\psi\rangle, |\varphi\rangle) = \cos\theta \||\psi\rangle\| \||\varphi\rangle\|, \tag{2.30}$$

onde θ é o ângulo entre $|\psi\rangle$ e $|\varphi\rangle$ a operação norma ($\|\|$) será definida mais adiante nessa seção. Se tratando de números complexos, dada a dificuldade de pensar no produto interno utilizando o ângulo entre vetores, se costuma utilizar um conceito diferente, apresentado na eq. (2.32). Inicialmente é importante saber que por simplicidade, ao utilizamos a notação de *Dirac*, nos referimos ao produto interno entre $|\psi\rangle$ e $|\varphi\rangle$, denotado anteriormente por $(|\psi\rangle, |\varphi\rangle)$, por um *braket* entre os vetores: $\langle\psi|\varphi\rangle$. Tal representação leva a notação de *Dirac* a também ser referenciada como notação *braket*. O *bra* de ψ , denotado por $\langle\psi|$, é o vetor dual do *ket* $|\psi\rangle$, definido como:

$$\langle\psi| = |\psi\rangle^\dagger = \begin{bmatrix} \psi_0 \\ \vdots \\ \psi_{n-1} \end{bmatrix}^\dagger = \begin{bmatrix} \psi_0^* & \dots & \psi_{n-1}^*, \end{bmatrix}, \tag{2.31}$$

onde a operação * denota o conjugado de um número complexo, e † (adaga) o conjugado transposto de um vetor.

O produto interno em $\mathbb{C}^n$ também pode ser definido como o produto de um *bra* por um *ket*:

$$(|\psi\rangle, |\varphi\rangle) = \langle\psi|\varphi\rangle = \begin{bmatrix} \psi_0^* & \psi_1^* & \dots & \psi_{n-1}^* \end{bmatrix} \begin{bmatrix} \varphi_0 \\ \varphi_1 \\ \dots \\ \varphi_{n-1} \end{bmatrix} = \sum_{i=0}^{n-1} \psi_i^* \varphi_i. \tag{2.32}$$

Dessa forma, o produto interno $\langle\psi|\varphi\rangle$ é o produto de $|\psi\rangle^\dagger$ e $|\varphi\rangle$.

Além disso, dado um espaço vetorial $\mathbb{C}^n$, $|\varphi\rangle$, $|\psi\rangle$ e $|\zeta\rangle \in \mathbb{C}^n$ e $\alpha \in \mathbb{C}$, o produto interno possui as seguintes propriedades:

Simetria Hermitiana

$$\langle\varphi|\psi\rangle = \langle\psi|\varphi\rangle\,; \tag{2.33}$$

Linearidade

$$\langle\varphi + \zeta|\psi\rangle = \langle\varphi|\psi\rangle + \langle\zeta|\psi\rangle\,; \tag{2.34}$$

Associatividade

$$\langle\alpha\varphi|\psi\rangle = \alpha\,\langle\varphi|\psi\rangle\,; \tag{2.35}$$

Positividade

$$\langle\varphi|\varphi\rangle \geq 0,$$

$$\langle\varphi|\varphi\rangle = 0 \Leftrightarrow \varphi = 0; \tag{2.36}$$

onde 0 é o vetor nulo.

Norma

A norma de um vetor é uma operação que retorna um escalar que representa seu tamanho, ela é definida por:

$$\||\varphi\rangle\| = \sqrt{\langle\varphi|\varphi\rangle} \geq 0. \tag{2.37}$$

Além disso, existe o conceito de um vetor normalizado, ou normal, isto é, um vetor com tamanho 1. Para normalizar um vetor, ou seja, manter sua direção no espaço vetorial, porém deixar seu tamanho igual a 1, podemos dividir o vetor por sua norma.

Ortogonalidade

Dois vetores são ditos ortogonais quando o produto interno deles é nulo.

$$\langle\varphi|\psi\rangle = 0 \Leftrightarrow |\varphi\rangle \perp |\psi\rangle\,. \tag{2.38}$$

Ortonormalidade

Um conjunto de vetores é dito ortonormal, quando todos os seus vetores são normais, e ortogonais dois a dois. Desta forma, com relação ao produto interno de quaisquer $|\varphi\rangle$ e $|\psi\rangle$ pertencentes ao conjunto temos que:

$$\langle\varphi|\psi\rangle = \begin{cases} 1, \text{ se } |\varphi\rangle = |\psi\rangle \\ 0, \text{ se } |\varphi\rangle \neq |\psi\rangle \end{cases} . \tag{2.39}$$

Produto Exterior

Como, pela notação de *Dirac*, $\langle\psi|$ é um vetor linha, podemos definir o produto exterior de $|\varphi\rangle$ e $|\psi\rangle$ como a multiplicação matricial $|\varphi\rangle\langle\psi|$:

$$\begin{aligned} |\varphi\rangle\langle\psi| &= \begin{bmatrix} \varphi_0 \\ \varphi_1 \\ \vdots \\ \varphi_{n-1} \end{bmatrix} \begin{bmatrix} \psi_0 & \psi_1 & \cdots & \psi_{n-1} \end{bmatrix} \\ &= \begin{bmatrix} \varphi_0\psi_0 & \varphi_0\psi_1 & \cdots & \varphi_0\psi_{n-1} \\ \varphi_1\psi_0 & \varphi_1\psi_1 & \cdots & \varphi_1\psi_{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_{n-1}\psi_0 & \varphi_{n-1}\psi_1 & \cdots & \varphi_{n-1}\psi_{n-1} \end{bmatrix} . \end{aligned} \tag{2.40}$$

2.2.4 Transformação Linear

Uma transformação linear é uma função entre dois espaços vetoriais. Dado T uma transformação linear que mapeia o espaço vetorial $\mathbb{C}^n$ para o espaço vetorial $\mathbb{C}^m$ podemos escrever $T : \mathbb{C}^n \to \mathbb{C}^m$, e temos as seguintes propriedades:

Preservação da soma

$$T(|\varphi\rangle + |\psi\rangle) = T\,|\varphi\rangle + T\,|\psi\rangle\,; \tag{2.41}$$

Preservação da multiplicação por escalar

$$T(\alpha \left|\varphi\right\rangle) = \alpha T \left|\varphi\right\rangle . \tag{2.42}$$

Além disso, quando o domínio e o contradomínio da transformação linear são iguais, ela também é chamada de operador linear.

Funcional linear

Um funcional linear é um tipo específico de transformação linear que mapeia um espaço vetorial à escalares. Por exemplo, dado $\left|\varphi\right\rangle$ e $\left|\psi\right\rangle \in \mathbb{C}^n$, um bra $\left\langle\varphi\right|$ pode ser definido como um funcional linear tal que $\left\langle\varphi\right| : \mathbb{C}^n \to \mathbb{C}$, desta forma, quando operado com $\left|\psi\right\rangle$ resulta em um escalar de valor complexo $\left\langle\varphi|\psi\right\rangle$.

Projeção

A projeção se utiliza do produto interno, então de maneira semelhante temos uma definição para quando se tratando de vetores reais e outra para vetores complexos. A projeção de $\left|\varphi\right\rangle$ em $\left|\psi\right\rangle$ (para $\left|\varphi\right\rangle, \left|\psi\right\rangle$ em $\mathbb{R}^2$ ou $\mathbb{R}^2$) retorna um vetor paralelo a $\left|\psi\right\rangle$ com tamanho proporcional ao tamanho de $\left|\varphi\right\rangle$ e o cosseno do ângulo entre os vetores.

Em $\mathbb{C}^n$, temos que, $\left\langle\psi|\varphi\right\rangle$ é o coeficiente de projeção de $\left|\varphi\right\rangle$ na direção $\left|\psi\right\rangle$. Então temos que $\left\langle\psi|\varphi\right\rangle \left|\psi\right\rangle$ é a projeção do vetor $\left|\varphi\right\rangle$ na direção $\left|\psi\right\rangle$, ou seja:

$$\text{proj}_{\left|\psi\right\rangle} \left|\varphi\right\rangle = \left\langle\psi|\varphi\right\rangle \left|\psi\right\rangle = \left|\psi\right\rangle\!\left\langle\psi\right| \left|\varphi\right\rangle , \tag{2.43}$$

aqui $\left|\psi\right\rangle\!\left\langle\psi\right|$ é um operador de projeção na direção $\left|\psi\right\rangle$.

A partir disso, derivamos a relação de completude que nos permite facilmente escrever qualquer vetor $\left|\varphi\right\rangle$ como a soma das suas projeções ortogonais com relação a uma base ortonormal. Por exemplo, dado $\beta =$

$\{|b_0\rangle, |b_1\rangle, \ldots, |b_{n-1}\rangle\}$ uma base ortonormal, podemos escrever $|\varphi\rangle$ como:

$$\begin{aligned} |\varphi\rangle &= \sum_{i=0}^{n-1} \text{proj}_{|b_i\rangle} |\varphi\rangle = \sum_{i=0}^{n-1} \langle b_i|\varphi\rangle |b_i\rangle \\ &= \sum_{i=0}^{n-1} |b_i\rangle\langle b_i| |\varphi\rangle = I |\varphi\rangle = |\varphi\rangle . \end{aligned} \tag{2.44}$$

2.2.5 Operadores Relevantes

Nesta seção, é apresentado um conjunto de operadores relevantes aos estudos da computação quântica.

Operadores Adjuntos

Dado um espaço de Hilbert V, o operador adjunto $A^\dagger$ (também chamado de conjugado Hermitiano) do operador A é tal que para todos os vetores $|\varphi\rangle, |\psi\rangle \in V$ vale:

$$(|\varphi\rangle, A|\psi\rangle) = \langle\varphi| A |\psi\rangle = (\langle\varphi| A) |\psi\rangle = (A^\dagger |\varphi\rangle, |\psi\rangle). \tag{2.45}$$

Seguindo a definição temos que $(AB)^\dagger = B^\dagger A^\dagger$, e $|\varphi\rangle^\dagger = \langle\varphi|$. Desta forma, temos que: $(A|\varphi\rangle)^\dagger = |\varphi\rangle^\dagger A^\dagger = \langle\varphi| A^\dagger$. Ao considerar a representação matricial do operador A, temos que sua relação com $A^\dagger$ é dada por: $A^\dagger = (A^*)^T$, o que possibilita observar que $(A^\dagger)^\dagger = A$.

Operadores Normais

Um operador A é dito normal se:

$$A^\dagger A = A A^\dagger, \tag{2.46}$$

além disso, pelo teorema da decomposição espectral temos que um operador é normal se e somente se é diagonalizável.

Operadores Autoadjuntos

Um operador A que tem como operador adjunto o próprio A é chamado de Autoadjunto (ou Hermitiano):

$$A^\dagger = A. \tag{2.47}$$

Por ter tal propriedade temos que um operador Hermitiano também é normal:

$$AA^\dagger = A^\dagger A = A^2, \tag{2.48}$$

além disso, seguindo o teorema da decomposição espectral de operadores hermitianos, temos que: um operador é Hermitiano se e somente se seus autovalores são reais.

Operadores Unitários

Um operador A é unitário se segue que:

$$AA^\dagger = A^\dagger A = I. \tag{2.49}$$

Dado que $AA^\dagger = A^\dagger A$, operadores unitários são automaticamente operadores normais. E pelo teorema da decomposição espectral de operadores unitários temos que dado um operador normal, ele é unitário se e somente se seus autovalores são números complexos módulo 1. Os operadores unitários se mostram úteis por preservar o produto interno entre vetores:

$$(A\left|\varphi\right\rangle, A\left|\psi\right\rangle) = \left\langle\varphi\right| A^\dagger A\left|\psi\right\rangle = \left\langle\varphi\right| I\left|\psi\right\rangle = \left\langle\varphi|\psi\right\rangle = (\left|\varphi\right\rangle, \left|\psi\right\rangle). \tag{2.50}$$

Operadores Positivos

Os operadores positivos são uma subclasse dos operadores Hermitianos, tal que, um operador A é positivo se para todo $\left|\varphi\right\rangle$:

$$\left\langle\varphi\right| A\left|\varphi\right\rangle \geq 0, \tag{2.51}$$

pelo teorema espectral temos que para todo operador positivo seus autovalores são reais (já que ele também é Hermitiano) e não negativos. Um

operador positivo pode também ser um operador positivo definido:

$$\langle\varphi|\, A\, |\varphi\rangle > 0, \tag{2.52}$$

para todo $|\varphi\rangle$ não nulo. Nesse caso, pelo teorema espectral temos que em todo operador positivo definido, seus autovalores são reais e positivos.

Operadores de Projeção

Um operador de projeção A leva o espaço vetorial V em algum subespaço W contido em V. Além disso, um operador de projeção A satisfaz:

$$A^2 = A. \tag{2.53}$$

Dado o subespaço vetorial k-dimensional W do espaço l-dimensional V $(l \geq k)$, é possível obter $\{|\varphi_0\rangle, \ldots, |\varphi_k\rangle, \ldots, |\varphi_l\rangle\}$, base ortonormal de V, de forma que $\{|\varphi_0\rangle, \ldots, |\varphi_k\rangle\}$ seja base ortonormal de W e o operador A de projeção para W seja definido por:

$$\mathrm{proj}_W = A = \sum_{i=0}^{k} |\varphi_i\rangle\langle\varphi_i|\,, \tag{2.54}$$

pela definição podemos ver que $|\varphi_i\rangle\langle\varphi_i|$ é Hermitiano:

$$(|\varphi_i\rangle\langle\varphi_i|)^\dagger = \langle\varphi_i|^\dagger\, |\varphi_i\rangle^\dagger = |\varphi_i\rangle\langle\varphi_i|\,, \tag{2.55}$$

portanto, A é Hermitiano e $A^\dagger = A$. Também temos que, para cumprir $A^2 = A$, a matriz diagonal de A também precisa ser um operador de projeção e portanto, os autovalores de A são 0 ou 1. E desta forma, todo operador de projeção é também, um operador positivo.

2.2.6 Produto Tensorial

O produto tensorial é uma operação que compõe um espaço vetorial a partir de outros dois espaços vetoriais. Essa operação é importante por permitir compor sistemas quânticos com mais de um qubit (seção 2.3.4).

Dado um espaço vetorial V de dimensão m e um espaço vetorial W de dimensão n, o espaço vetorial V tensorial W (denotado por $V \otimes W$) possui dimensão nm. E um elemento de $V \otimes W$ é adquirido através do produto tensorial $|v\rangle \otimes |w\rangle$, onde, $|v\rangle \in V$ e $|w\rangle \in W$. Além disso podemos usar as seguintes notações para $|v\rangle \otimes |w\rangle$: $|v\rangle |w\rangle$, $|v, w\rangle$ e $|vw\rangle$.

Temos também que, dados $|v\rangle, |v'\rangle \in V$, $|w\rangle, |w'\rangle \in W$ e o escalar α o produto tensorial possui as seguintes propriedades:

$$\alpha(|v\rangle \otimes |w\rangle) = (\alpha |v\rangle) \otimes |w\rangle = |v\rangle \otimes (\alpha |w\rangle); \tag{2.56}$$

$$(|v\rangle + |v'\rangle) \otimes |w\rangle = |v\rangle \otimes |w\rangle + |v'\rangle \otimes |w\rangle ; \tag{2.57}$$

$$|v\rangle \otimes (|w\rangle + |w'\rangle) = |v\rangle \otimes |w\rangle + |v\rangle \otimes |w'\rangle . \tag{2.58}$$

Agora que temos um novo espaço vetorial, precisamos de novos operadores lineares para operarmos sobre $V \otimes W$. Dado um operador A sobre V e um operador B sobre W, podemos definir o operador $A \otimes B$ sobre $V \otimes W$:

$$(A \otimes B)(|v\rangle \otimes |w\rangle \equiv A |v\rangle \otimes B |w\rangle , \tag{2.59}$$

podemos visualizar melhor o operador $A \otimes B$ na representação do produto de *Kronecker*. Dada as matrizes $A_{m\times n}$ e $B_{p\times q}$, $A \otimes B$ é uma matriz $mp \times nq$:

$$A \otimes B \equiv \underbrace{\left.\begin{bmatrix} A_{11}B & A_{12} & \dots & A_{1n}B \\ A_{21}B & A_{22} & \dots & A_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1}B & A_{m2} & \dots & A_{mn}B \end{bmatrix}\right\} m \times p}_{n\times q} \tag{2.60}$$

Outra notação usada é $|\psi\rangle^{\otimes n}$, quando fazemos produto tensorial sobre o elemento $|\psi\rangle$ n vezes:

$$\begin{aligned} |\psi\rangle^{\otimes n} &= |\psi\rangle^{\otimes n-1} \otimes |\psi\rangle\,, \\ |\psi\rangle^{\otimes 2} &= |\psi\rangle \otimes |\psi\rangle\,. \end{aligned} \tag{2.61}$$

Dada a revisão dos conceitos de álgebra linear, agora serão apresentados os conceitos da mecânica quântica.

2.3 Postulados da Mecânica Quântica

A computação quântica se alicerça na mecânica quântica para computar, portanto é fundamental o entendimento de seus conceitos mais básicos, antes de partir à computação em si.

A mecânica quântica, por si, não descreve as leis aos quais os sistemas físicos estão sujeitos, porém, disponibiliza o aparato matemático e conceitual para desenvolvimento dessas leis.

Nielsen e *Chuang* [44] enunciam os postulados da mecânica quântica, eles são referentes a: Espaço de um Sistema (seção 2.3.1), Evolução de um Sistema (seção 2.3.2), Medida de um Sistema (seção 2.3.3), Sistema Composto (seção 2.3.4).

2.3.1 Estado de um Sistema

Postulado 1. *A todo sistema físico fechado existe um espaço de Hilbert associado. O estado do sistema pode ser completamente descrito por um vetor unitário pertencente a esse espaço.*

O primeiro postulado diz sobre como a mecânica quântica descreve um sistema físico isolado, ou seja, um sistema que não tem nenhum tipo de contato com o ambiente externo, como troca de energia, por exemplo. Por outro lado, não descreve qual é, ou como descobrir, qual é o espaço de Hilbert, e o vetor unitário associado a cada sistema físico. Desta forma, é

necessário um estudo para cada sistema, a fim de descobrir o seu espaço associado e o vetor que descreve seu estado.

Na computação quântica, trataremos de sistemas isolados - os computadores quânticos. Desta forma, neste trabalho trataremos o estado dos modelos apresentados como um vetor unitário pertencente a um espaço de Hilbert.

Qubit - O bit quântico

O qubit é o exemplo mais simples de um sistema quântico fechado, mas antes de entender o que é um qubit, é interessante entender sua versão clássica, o bit. Em um sistema computacional clássico o bit é o menor pedaço de informação existente. Ele possui dois estados possíveis ou interpretações, como por exemplo: ligado ou desligado, "0" ou "1" e verdadeiro ou falso. Combinando mais bits podemos ter mais estados e, portanto, mais interpretações. Dessa forma, podemos guardar mais informação e processa-la de mais formas. A quantidade de informações diferentes armazenada em um conjunto de bits cresce de maneira exponencial, ou seja, ao adicionar um bit é possível representar o dobro de informação.

O bit quântico

O bit quântico, qubit, [44] é um conceito análogo ao bit clássico. Ele representa o menor pedaço de informação que um sistema quântico pode representar. Os qubits, neste trabalho, serão objetos matemáticos, mas assim como os bits podem ser interpretados em sistemas físicos de formas variadas e implementados de diferentes formas.

Um qubit, assim como um bit, possui 2 estados como por exemplo $|0\rangle$ e $|1\rangle$, análogos ao "0" e "1" clássicos. Porém, existem mais estados possíveis para um qubit. Um qubit $|\psi\rangle$ arbitrário pode estar em uma combinação linear dos estados, uma superposição; desta forma, $|\psi\rangle$ é descrito por:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{2.62}$$

onde α e β são valores que representam a amplitude das projeções de $|\psi\rangle$ nos estados $|0\rangle$ e $|1\rangle$, respectivamente. De forma simplificada, representam a proporção de $|\psi\rangle$ nos estado $|0\rangle$ e $|1\rangle$. Em geral α e β serão números complexos. $|0\rangle$ e $|1\rangle$ são vetores colunas de uma base ortonormal que representam os estados análogos aos estados "0" e o "1" do bit.

Por se tratar de um sistema quântico, a mecânica quântica nos garante não termos acesso ao estado atual de um qubit, isto é, aos valores α e β, tão facilmente quanto ao estado de um bit, que pode ser acessado a qualquer momento, quantas vezes quisermos e sem perda de informação. Ao ler um bit, as únicas respostas possíveis são "0" ou "1". De forma similar, no caso de um qubit temos que até antes de um qubit ser medido (o equivalente a ler um bit) ele pode estar em uma superposição dos seus estados, porém, ao ser medido as únicas respostas possíveis são os estados base, nesse caso, $|0\rangle$ e $|1\rangle$. Portanto, após a medida o estado colapsa para uma dessas possibilidades e qualquer leitura feita nele, enquanto ele não for operado novamente, sempre retornará o mesmo resultado. A mecânica quântica nos garante que no momento da medida teremos como resultado da leitura de $|\psi\rangle$, $|0\rangle$ com probabilidade $|\alpha|^2$ ou $|1\rangle$ com probabilidade $|\beta|^2$, onde $|\alpha|^2$ e $|\beta|^2$ somam 1, como manda a probabilidade. Por causa disto, lidamos sempre com nossos estados com tamanho '1', ou seja, um vetor normalizado.

Representações de um Qubit

Por termos $|\alpha|^2 + |\beta|^2 = 1$, é possível reescrever o estado $|\psi\rangle$ da eq. (2.62) da seguinte maneira:

$$|\psi\rangle = e^{i\gamma}(\cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle), \tag{2.63}$$

onde γ, θ e φ são números reais. É possível provar que o fator $e^{i\gamma}$, mesmo correspondendo a uma alteração da fase global, não possui um efeito observável, portanto a eq. (2.63) pode ser reescrita como:

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle, \tag{2.64}$$

onde θ e φ são números que especificam o ângulo polar e azimutal, respectivamente, de forma a, juntos, definirem um ponto na superfície de uma esfera 3-dimensional de raio 1. Essa esfera representa todas as possibilidades de estados que um qubit pode ter, e é chamada de esfera de Bloch (fig. 10).

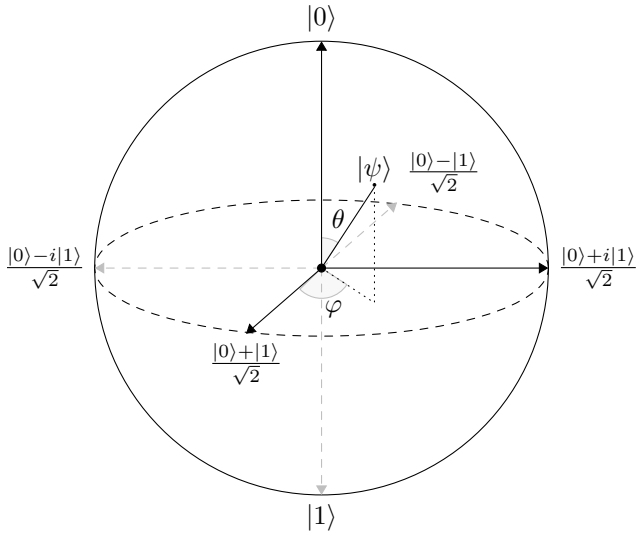

Figura 10 – Representação de um qubit na esfera de Bloch.

Na esfera temos que $|0\rangle$ se encontra ao norte do eixo z e $|1\rangle$ ao sul; variando o ângulo azimutal variamos tanto no eixo x, entre $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ e $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$, quanto no eixo y, entre $\frac{|0\rangle+i|1\rangle}{\sqrt{2}}$ e $\frac{|0\rangle-i|1\rangle}{\sqrt{2}}$. A esfera de Bloch é uma abstração útil para visualização de 1 qubit por permitir interpretar as variações do estado de um qubit, como rotações aplicadas com relação aos ângulos θ e φ.

Um qubit pode estar em qualquer ponto da esfera de Bloch, portanto possui infinitas possibilidades de estados, o que pode facilmente nos fazer pensar que um qubit pode armazenar informação infinita. Porém, como ao medir um qubit só temos duas possíveis respostas, não temos realmente informação infinita armazenadas em um qubit. Por outro lado, antes da

medida, realmente podemos ter mais informação do que $|0\rangle$ e $|1\rangle$ como veremos posteriormente.

2.3.2 Evolução de um Sistema

Postulado 2. *A evolução no tempo do estado de um sistema quântico fechado é dada pela aplicação de um operador unitário. Portanto a evolução do estado $|\psi_0\rangle$, no tempo 0, para o estado $|\psi_1\rangle$, no tempo 1, é descrita por um operador unitário U tal que:*

$$|\psi_1\rangle = U\,|\psi_0\rangle\,, \tag{2.65}$$

A mecânica quântica, através do segundo postulado, nos diz qual é a relação entre dois estados de um sistema quântico em diferentes instantes no tempo. E que essa relação é a aplicação de um operador unitário. Porém, ela não diz como é o operador que descreve a evolução em tal sistema. Ela só garante que a evolução no tempo de um sistema é descrito desta forma.

Portanto, em um computador quântico, onde o tempo é discreto, cada variação por unidade de tempo - um passo de computação - é descrito pela aplicação de um operador unitário.

2.3.3 Medida de um Sistema

Postulado 3. *Uma medida quântica pode ser descrita por um conjunto de operadores de medida $\{M_m\}$ que operam sobre o estado associado ao sistema. Onde o índice m referencia um possível resultado da medida. Portanto, dado um estado $|\psi\rangle$ em um sistema quântico, ao tirar a medida a probabilidade do resultado m ocorrer é:*

$$p(m) = \langle\psi|\, M_m^\dagger M_m\,|\psi\rangle\,, \tag{2.66}$$

e o estado do sistema após a medida é dado por:

$$\frac{M_m\,|\psi\rangle}{\sqrt{p(m)}} = \frac{M_m\,|\psi\rangle}{\sqrt{\langle\psi|\, M_m^\dagger M_m\,|\psi\rangle}}. \tag{2.67}$$

Os operadores de medida satisfazem a equação de completude:

$$\sum_m M_m^\dagger M_m = I, \tag{2.68}$$

de maneira equivalente, temos:

$$1 = \sum_m p(m) = \sum_m \langle\psi| M_m^\dagger M_m |\psi\rangle . \tag{2.69}$$

Dado um sistema quântico fechado o postulado 2 descreve como é dada sua evolução. Porém em dados momentos, precisamos observar o sistema para saber o que ocorre internamente. Essa interação faz com que o sistema não esteja mais isolado, e portanto, a partir de agora, sua evolução pode não ser descrita por evoluções unitárias. O postulado 3 descreve o que ocorre quando um sistema quântico fechado é medido.

Dado um qubit na base computacional, por exemplo, temos que seus operadores de medida são:

$$M_0 = |0\rangle\langle 0| , \tag{2.70}$$

$$M_1 = |1\rangle\langle 1| , \tag{2.71}$$

como tanto M_0 como M_1, são projeções hermitianas, temos que:

$$M_0^\dagger M_0 = M_0^2 = M_0, \tag{2.72}$$

$$M_1^\dagger M_1 = M_1^2 = M_1. \tag{2.73}$$

E a equação de completude é obedecida:

$$\sum_{m=0}^{1} M_m^\dagger M_m = \sum_{m=0}^{1} M_m = I. \tag{2.74}$$

Supondo que em dado momento da computação, o sistema esteja no estado $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. A probabilidade de se medir $|0\rangle$ e $|1\rangle$ é:

$$p(0) = \langle\psi| M_0^\dagger M_0 |\psi\rangle = \langle\psi| M_0 |\psi\rangle = |\alpha|^2, \tag{2.75}$$

$$p(1) = \langle\psi| M_1^\dagger M_1 |\psi\rangle = \langle\psi| M_1 |\psi\rangle = |\beta|^2. \tag{2.76}$$

Após os respectivos resultados, o sistema se encontra no seguinte estado:

$$\frac{M_0 |\psi\rangle}{|\alpha|} = \frac{\alpha}{|\alpha|} |0\rangle , \tag{2.77}$$

$$\frac{M_1 |\psi\rangle}{|\beta|} = \frac{\beta}{|\beta|} |1\rangle . \tag{2.78}$$

2.3.4 Sistema Composto

Postulado 4. *O estado de um sistema físico composto é o produto tensorial dos estados de Hilbert associados às componentes do sistema, ou seja, um sistema composto por n sistemas quânticos - nos estados* $|\psi\rangle_0$, $|\psi\rangle_1$, ..., $|\psi\rangle_{n-1}$ *- tem como estado* $|\psi\rangle_0 \otimes |\psi\rangle_1 \otimes \ldots \otimes |\psi\rangle_{n-1}$

O quarto postulado nos diz como lidar com sistemas compostos de múltiplos qubits. No caso de 2 qubits na base computacional, cada qubit pertence a $\mathbb{C}^2$ e o sistema completo, pertence a $\mathbb{C}^{\otimes 2}$, temos as seguintes possibilidades de estado:

$$\begin{aligned} |00\rangle = |0\rangle \otimes |0\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} ; |01\rangle = |0\rangle \otimes |1\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} ; \\ |10\rangle = |1\rangle \otimes |0\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} ; |11\rangle = |1\rangle \otimes |1\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} . \end{aligned} \tag{2.79}$$

Para definir um sistema em $\mathbb{C}^{\otimes n}$, portanto com n qubits, podemos usar a notação da base computacional, e colocar valores entre 0 e $2^n - 1$ dentro do ket:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} , |1\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} , |2\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} , \ldots , |2^n - 2\rangle = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \end{bmatrix} , |2^n - 1\rangle = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 1 \end{bmatrix} , \tag{2.80}$$

onde neste caso, todos esses vetores tem 2^n linhas.

Evolução de um sistema composto

Seguindo o postulado 2 (seção 2.3.2), a evolução de um sistema composto segue pela aplicação de um operador unitário. Em um sistema composto de n qubits os operadores em questão serão matrizes de $2^n \times 2^n$. Um operador de evolução para um sistema composto pode ser obtido, por exemplo, através do produto tensorial entre os operadores de cada sistema:

$$U = U_0 \otimes U_1 \otimes \ldots \otimes U_{n-1}, \tag{2.81}$$

onde U_i opera com relação ao i-ésimo qubit. Neses casos, onde é possivel compor a operação U por meio do produto tensorial de operações de 1 qubit, podemos facilmente operar somente com os qubits desejáveis, digamos o i-ésimo e o k-ésimo qubit da seguinte forma:

$$U = I_0 \otimes \ldots \otimes I_{i-1} \otimes U_i \otimes I_{i+1} \otimes \ldots \otimes I_{k-1} \otimes U_k \otimes I_{k+1} \otimes \ldots \otimes I_{n-1}. \tag{2.82}$$

Outra notação que pode ser usada neste caso é:

$$U = U_i \otimes U_k, \tag{2.83}$$

onde neste caso entende-se que U_i será aplicado somente com relação ao i-ésimo qubit e U_k será aplicado somente com relação ao i-ésimo qubit, e todos os outros se mantêm inalterados.

Medida de um sistema composto

De forma similar a evolução de um sistema composto, os operadores de medida também são matrizes $2^n \times 2^n$. E podem ser obtidos pelo produto tensorial dos operadores de medida dos bits desejados. Seja um $|\psi\rangle$ um sistema composto de n qubits na base computacional:

$$|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle + \ldots + \alpha_{2^n-1} |2^n - 1\rangle ; \tag{2.84}$$

os operadores para medir o seu i-ésimo qubit podem ser definidos por:

$$\begin{aligned} M_{i0} &= I_0 \otimes I_1 \otimes \ldots \otimes |0\rangle\langle 0|_i \otimes \ldots \otimes I_{2^n-1}, \\ M_{i1} &= I_0 \otimes I_1 \otimes \ldots \otimes |1\rangle\langle 1|_i \otimes \ldots \otimes I_{2^n-1}, \end{aligned} \tag{2.85}$$

onde M_{i0} é o operador que mede o i-ésimo qubit com relação ao valor 0, e M_{i1} é o operador que mede o i-ésimo qubit com relação ao valor 1.

A probabilidade de se obter o resultado 0 no i-ésimo qubit é:

$$p_i(0) = \langle\psi| M_{i0}^\dagger M_{i0} |\psi\rangle , \tag{2.86}$$

e o estado do sistema após a medida é:

$$|\psi'\rangle = \frac{M_{i0}|\psi\rangle}{\sqrt{p_i(0)}}. \tag{2.87}$$

De maneira análoga, no caso do resultado 1 temos a seguinte probabilidade:

$$p_i(1) = \langle\psi| M_{i1}^\dagger M_{i1} |\psi\rangle , \tag{2.88}$$

e o estado do sistema após a medida é:

$$|\psi'\rangle = \frac{M_{i1}|\psi\rangle}{\sqrt{p_i(1)}}. \tag{2.89}$$

Estados emaranhados

É possível que um sistema composto não possa ser decomposto num produto tensorial de seus qubits, neste caso o sistema está em um estado emaranhado. A principal característica de um estado emaranhado é que existem qubits correlacionados e a leitura de um deles pode fazer com que um outro também colapse. É impossível definir o estado de cada qubit para o estado emaranhado, por exemplo o seguinte estado:

$$|\beta_{00}\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}. \tag{2.90}$$

Por não podermos decompor $|\beta_{00}\rangle$ em um produto tensorial de 2 qubits, dizemos que ele está emaranhado, e seus qubits correlacionados. Ao tomar

a medida do primeiro qubit, podemos obter o resultado $|0\rangle_0$ com probabilidade $\frac{1}{2}$, e resultado $|1\rangle_0$ também com probabilidade $\frac{1}{2}$. Supondo $|0\rangle_0$ como resultado da medida, o segundo qubit também colapsara para $|0\rangle_1$, e o sistema por inteiro estará em $|00\rangle$. De forma análoga, se $|1\rangle_0$ for o resultado da medida, o segundo qubit também colapsara para $|1\rangle_1$, e o sistema por inteiro estará em $|11\rangle$.

$|\beta_{00}\rangle$ é um dos quatro estados de *Bell*, que são um conjunto de estados emaranhados que definem uma base de estados emaranhados de 2 qubits:

$$\begin{aligned} |\beta_{00}\rangle &= \frac{|00\rangle + |11\rangle}{\sqrt{2}}; \\ |\beta_{01}\rangle &= \frac{|00\rangle - |11\rangle}{\sqrt{2}}; \\ |\beta_{10}\rangle &= \frac{|01\rangle + |10\rangle}{\sqrt{2}}; \\ |\beta_{11}\rangle &= \frac{|01\rangle - |10\rangle}{\sqrt{2}}. \end{aligned} \tag{2.91}$$

Feita a revisão teórica necessária, o capítulo seguinte apresenta os estudos realizados no presente trabalho.

Capítulo 3

Modelos Computacionais Quânticos

Neste capítulo se apresentarão algumas versões quânticas dos autômatos finitos e de pilha indicando algumas relações entre eles e avanços conhecidos (seção 3.1) e, também, apresentará de maneira mais detalhada um dos modelos (seção 3.2).

3.1 Histórico dos Autômatos

Deutsch [18] foi o primeiro a considerar um autômato quântico ao reavaliar a Tese de *Church-Turing* considerando os princípios da mecânica quântica e com isso propôs uma versão quântica da Máquina de Turing, além disso, *Deutsch* também demonstrou a existência de uma Máquina de Turing Quântica universal. Após *Deutsch* vários outros autômatos foram propostos.

Esta seção traz em ordem histórica alguns dos modelos existentes, introduz resumidamente cada um, avanços conhecidos, além de algumas relação entre eles e suas versões clássicas. Esta seção se baseia principalmente nos trabalhos de *Bhatia* e *Kumar* [15], *Ambainis* e *Yakaryılmaz* [11], *Qiu* e *Li et al* [59] e *Qiu* e *Li* [49].

3.1.1 MM-1QFA

Tratando-se de autômato finito quântico(QFA) *Kondacs* e *Watrus* [32] foram os primeiros a desenvolverem um modelo, hoje tal modelo é conhecido por *Measure Many one-way quantum finite automata*(MM-1QFA). O MM-1QFA foi introduzido em 1997, e atualmente é chamado assim graças a algumas de suas características: em todo passo da computação uma medida é feita(*Measure Many*). Dado o resultado, o autômato pode aceitar, rejeitar ou continuar computando; além disso, o cabeçote de leitura sempre se move para direita, o que, em contrapartida a versões onde o cabeçote possui mais liberdade de movimentação, o caracteriza como *one-way*.

Quando definido originalmente [32], o MM-1QFA possuía marcadores de início e fim de fita, porém em 2002, *Brodsky* e *Pippenger* [16] demonstraram como remover o marcador de início de fita sem diminuir o poder do autômato.

Kondacs e *Watrus* [32] quando introduziram o MM-1QFA provaram que com erro limitado ele reconhece um subconjunto próprio das linguagens regulares (o conjunto das linguagens reconhecidas por AF's) [58] . *Yakaryilmaz* e *Say* [61] [62] provaram em 2009, que o MM-1QFA com erro ilimitado reconhece a classe das linguagens estocásticas, a classe das linguagens reconhecidas pelos autômatos finitos probabilísticos [52]. Por outro lado, eles [61] também demonstraram que com o conjunto das linguagens estocásticas é um conjunto próprio das linguagens reconhecidas pelo MM-1QFA com ponto-de-corte-0.

Brodsky e *Pippenger* [16], em 2002, demonstraram que as classes de problemas resolvidos por MM-1QFA com erro limitado e com erro ilimitado são fechadas no complemento [58], homomorfismo inverso [55] e *word quotient* [55], eles também provaram que ambas as classes não são fechadas no homomorfismo [55]. Em 2001, *Ambainis* e *Kikusts et al* [7] provou o não fechamento do MM-1QFA com relação às operações booleanas.

Ambainis e *Freivalds* [5] demonstraram também que o MM-1QFA aceita linguagens com probabilidade maior que $\frac{7}{9}$ se e somente se essa linguagem

também for aceita por um autômato finito reversível. De forma similar, *Ambainis* e *Ķikusts* [6] provaram em 2003 que, toda linguagem não reconhecida por um autômato finito reversível é reconhecida com probabilidade máxima de 0.7726 pelo MM-1QFA. *Kikusts* [31], *Ambainis* e *Freivalds* [5] demonstram que para determinadas linguagens é possível construir MM-1QFA com uma redução quadrática e exponencial, respectivamente, no número de estados quando comparados as alternativas clássicas e probabilísticas. Por outro lado, *Ambainis* e *Nayak et al* [8] também demonstraram que existem linguagens para as quais o MM-1QFA precisa de um número exponencial de estados a mais que o AF mínimo.

3.1.2 2QFA

De maneira similar ao MM-1QFA, *Kondacs* e *Watrus* [32] consideraram também um autômato de 2 direções, com capacidade de movimentar o cabeçote para direita, esquerda ou não movê-lo. Este modelo é chamado de *two-way quantum finite automata*(2QFA). Essa característica torna o 2QFA mais complexo para o desenvolvimento e também mais poderoso que o MM-1QFA. O 2QFA reconhece as linguagens regulares [32], algumas linguagens livres de contexto e algumas linguagens não livre de contexto com erro unilateral em tempo linear [32].

3.1.3 1.5-QFA

Em 1999, *Amano* e *Iwama* [3] propuseram o 1.5-way quantum finite automata(1.5-QFA), capaz de mover o cabeçote para direita ou não movê-lo.

3.1.4 MO-1QFA

Em 2000 *Moore* e *Crutchfield* [39] aplicaram diretamente os princípios da mecânica quântica, e propuseram o modelo mais simples de autômato finito quântico, atualmente referenciado por *measure once one-way quan-*

tum finite automata(MO-1QFA), sua principal diferença do MM-1QFA é que a medida só ocorre ao final da computação. Vários estudos foram feitos com relação a classe de linguagens reconhecidas por MO-1QFA. *Moore* e *Crutchfield* [39] provaram também que a classe de linguagens reconhecidas pelo MO-1QFA é fechada sobre adição, multiplicação, complemento e homomorfismo inverso. Além disso, *Moore* e *Crutchfield* [39] definiram um lema do bombeamento para o conjunto de linguagens reconhecidas pelo MO-1QFA.

Bertoni e *Carpentieri* [13] demonstraram em 2001 que a classe de linguagens reconhecidas pelo MO-1QFA com erro limitado é a mesma reconhecida pelos autômatos finitos reversíveis. Em 2002, *Brodsky* e *Pippenger* [16] provaram que a classe de linguagens reconhecida pelo MO-1QFA com erro limitado é a mesma reconhecida por *group automata*, um subconjunto próprio das linguagens regulares. Por outro lado, *Qiu* e *Ying* [50] e *Brodsky* e *Pippenger* [16] provaram que com erro ilimitado é possível reconhecimento de algumas linguagens não regulares e até não livres de contexto. De toda forma, *Amano* e *Iwama* [3] mostraram que mesmo com erro ilimitado o MO-1QFA ainda reconhece um subconjunto das linguagens estocásticas (linguagens reconhecidas pelo autômato finito probabilístico com ponto de corte). *Bertoni* e *Carpentieri* [13] mostraram também que o MO-1QFA não reconhece nenhuma linguagem finita.

Além disso, *Brodsky* e *Pippenger* [16], *Koshiba* [33] e *Li* e *Qiu* [34] definiram algoritmos para definir a equivalência entre MO-1QFA em tempo polinomial.

3.1.5 MC-QPDA

No mesmo *paper* em que *Moore* e *Crutchfield* [39] propuseram o MO-1QFA eles também propuseram a primeira versão quântica de um autômato de pilha, atualmente o modelo é chamado de MC-QPDA como referência aos autores. Esse modelo foca na generalização do autômato de pilha, dessa forma, os operadores de evolução não são nem unitários, nem

isométricos.

3.1.6 AQFA

Paschen [45] propôs em 2000, adicionar qubits ancilas ao processamento para reduzir a restrição de que as transições fossem unitárias e, dessa forma, ampliando seu poder computacional, tal modelo é referenciado como *ancilla quantum finite automata*(AQFA). A adição dos qubits ancilas são feitas adicionando um alfabeto de saída. Foi provado também [45], que para toda linguagem regular existe uma quantidade de qubits ancilas com o qual o AQFA reconhece linguagem de maneira exata.

3.1.7 Go-QPDA

Baseado no MC-QPDA, *Golovkins* [23] introduziu em 2000 sua versão de autômato de pilha, referenciado por Go-QPDA em nome do autor. Nele operações de evolução devem ser unitárias e sua função de transição exige condições de boa formação. Além disso, o Go-QPDA reconhece algumas linguagens não reconhecidas por autômatos de pilha, e por autômatos de pilha probabilísticos.

3.1.8 Gu-QPDA

Também em 2000, *Gudder* [25], apresentou um modelo de autômato de pilha, o Gu-QPDA, outra generalização dos autômatos de pilha onde os operadores são isométricos, porém não unitários.

3.1.9 2QCFA

Em 2002, *Ambainis* e *Watrous* [9] propuseram um modelo com processamento quântico e clássico, o *two-way quantum finite automata with quantum and classical states*(2QCFA). Esse modelo possui tanto estados quânticos quanto clássicos, além de dois conjuntos de transições, um que lida com as evoluções quânticas, operações unitárias e medidas, e outro

que descreve as transições dos estados clássicos. Nesse modelo, a parte quântica age como um controle, decidindo como será a leitura da fita, movendo-a para a direita, para a esquerda ou não movendo, enquanto a parte clássica realiza a interação com a entrada. *Ambainis* e *Watrous* [9] provaram que o 2QCFA é mais poderoso que os autômatos probabilísticos, reconhecendo o conjunto das linguagens regulares com exatidão, além de reconhecer algumas linguagens não regulares como: palíndromos constituídos e a's e b's e palavras com mesma quantidade de a's e b's. Em 2008, *Qiu* [48] verificou que o 2QCFA é fechado com relação às operações Booleanas, e que a concatenação é fechada dependendo das restrições aplicadas, por exemplo, ao possibilitar erro unilateral.

3.1.10 CL-1QFA

Em 2003,*Bertoni* e *Mereghetti et al* [14] propuseram o *quantum automata with control language*(CL-1QFA), um modelo similar ao MM-1QFA, que possibilita projeções de medida arbitrárias por todo espaço gerado pelos estados da base, que utiliza a linguagem alvo como controle para definir se deve aceitar ou não a entrada. *Bertoni* e *Mereghetti et al* [14] também demonstraram que o CL-1QFA com erro limitado é fechado nas operações booleanas, diferentemente do MM-1QFA e que o CL-1QFA com erro limitado reconhece é um conjunto próprio das linguagens regulares. *Li* e *Qiu* [35] consideraram também o problema da equivalência entre 2 CL-1QFA e definiram um algoritmo polinomial para isso.

3.1.11 LQFA

Em 2004, *Ambainis* e *Beaudry et al* [4], propuseram o *latvian quantum finite automata*(LQFA), uma generalização do MO-1QFA, onde a função de transição inclui tanto medidas de projeção quanto operações unitárias.

3.1.12 Na-QPDA

Ainda em 2004, *Nakanishi* [43] propôs um autômato de pilha quântico com pilha clássica. Foi provado também que o Na-QPDA reconhece algumas linguagens não livres de contexto com erro unilateral [43].

3.1.13 qQPDA

Qiu e *Ying* [50] propuserem em 2004 o *q quantum pushdown automata*. Toda linguagem reconhecida pelo MC-QPDA por pilha vazia é também reconhecida pelo qQPDA, além disso, comparado ao Go-QPDA, o qQPDA possui uma função de transição mais simplificado [49].

3.1.14 1QFAC

Em 2009,*Qiu* e *Mateus et al* [51] propuseram o *one-way quantum finite automata together with classical states* (1QFAC), um modelo similar ao 2QCFA com estados quânticos e clássicos. Onde a parte clássica do autômato funciona como um controle que determina qual transformação será feita no estado quântico para cada símbolo da entrada. Porém, diferente do 2QCFA seu cabeçote de leitura só se move para direita e como no MO-1QFA, sua medida só ocorre ao final da entrada.

Qiu e *Mateus et al* [51] mostraram também que o 1QFAC com erro limitado reconhece exatamente o conjunto das linguagens regulares. Provaram também a existência de linguagens não reconhecidas tanto pelo MM-1QFA quanto pelo MO-1QFA, porém reconhecidas por um AF com $O(m)$ estados, e reconhecida por um 1QFAC com 2 estados clássicos e $O(log(m))$ estados quânticos.

Qiu e *Mateus et al* [51] também estudaram sobre a equivalência entre 2 1QFAC, definindo um algoritmo que em tempo polinomial para determinar essa equivalência.

3.1.15 MO-1GQFA

Em 2010, *Hirvensalo* [27] desenvolveu um modelo mais genérico a partir do MO-1QFA, onde a função transição ao invés de ser unitária preserva o mapeamento do traço positivo [44]. Atualmente ele é conhecido como *measure once one-way general quantum finite automata*(MO-1GQFA).

3.1.16 MM-1GQFA

O mesmo princípio utilizado no MO-1GQFA foi aplicado por *Li* e *Qiu et al* [36] em 2012 para generalizar o MM-1QFA, criando o *measure many one-way quantum finite automata*(MM-1GQFA).

A tabela 2 sumariza as características dos modelos apresentados nesta seção. Em seguida a fig. 11 mostra uma comparação do poder computacional dos autômatos finitos.

MM-1QFA	Múltiplas medidas, cabeçote se move em uma direção, reconhece o conjunto das linguagens estocásticas utilizando ponto-de-corte-0. Propriedades fechadas conhecidas: complemento, homomorfismo inverso, *word quotient*. Propriedades não fechadas conhecidas:homomorfismo e operações booleanas.
2QFA	Múltiplas medidas, cabeçote pode não se mover ou se move para direita e esquerda. É conhecido por ser o modelo mais poderoso entre os autômatos finitos .

Tabela 2 – Resumo das características relevantes dos autômatos.

1.5-QFA	Cabeçote pode não se mover ou ir para direita.
MO-1QFA	Medida ocorre somente ao final da computação, cabeçote sempre se move para direita. É a versão menos poderosa de AF. Propriedades fechadas conhecidas: adição, multiplicação, complemento e homomorfismo inverso. Reconhece um subconjunto próprio das linguagens estocásticas.
MC-QPDA	Generaliza o PDA, suas transições se baseiam em operadores de evolução não unitários e não isométricos.
AQFA	Utiliza qubits ancilares. Sempre existe um AQFA que pode reconhecer uma linguagem regular de maneira exata.
Go-QPDA	Suas evoluções são unitárias e bem formadas. Reconhece algumas linguagens não reconhecidas por autômatos de pilha estocástico.
Gu-QPDA	Suas evoluções são isométricas, porém não unitárias.

Tabela 2 – Resumo das características relevantes dos autômatos.

2QCFA	Possui tanto estados clássicos quanto quânticos. Reconhece o conjunto das linguagens regulares com exatidão. Propriedades fechadas conhecidas: operações booleanas, e concatenação(dependendo de algumas restrições).
CL-1QFA	Possibilita projeções de medidas arbitrárias. Propriedades fechadas conhecidas: operações booleanas. Com erro limitado reconhece um conjunto próprio das linguagens regulares.
LQFA	Sua função de transição inclui medidas de projeção e operadores unitários.
Na-QPDA	É um autômato quântico, porém com pilha clássica. Reconhece linguagens não livres de contexto com erro unilateral.
qQPDA	Reconhece toda linguagem reconhecida pelo MC-QPDA por pilha vazia.
1QFAC	Possui estados clássicos e quânticos. Com medida somente ao final da computação e um cabeçote que sempre se move para direita. Com erro limitado reconhece o conjunto das linguagens regulares.

Tabela 2 – Resumo das características relevantes dos autômatos.

MO-1GQFA	É uma generalização do MO-1QFA que preserva o traço positivo na sua função de transição.
MM-1GQFA	É uma generalização do MM-1QFA que preserva o traço positivo na sua função de transição.

Tabela 2 – Resumo das características relevantes dos autômatos.

Feita a apresentação dos modelos e trabalhos até então, o presente trabalho se debruça sobre MO-1QFA, sua definição, algumas linguagens específicas e comparações com sua versão clássica.

3.2 *Measure-Once Quantum Finite Automata*

Como visto na seção anterior o MO1QFA é o modelo mais simples de autômato finito quântico e que possui a menor expressividade. Esse modelo foi proposto por *Moore C.* e *Crutchfield J.* [39] em 2000, com o intuito de adicionar os efeitos quânticos ao autômato finito, tais como: transições dadas por transformações unitárias e a necessidade de se medir o sistema para obter alguma informação. A sigla MO1QFA vêm de *measure-once 1-way quantum finite automata*, nome obtido dado algumas características do seu funcionamento:

- o estado quântico é medido uma única vez e sempre ao final da execução, por isto é chamado de *measure-once*;
- em comparação a outras versões de autômatos que podem mover o cabeçote de leitura para ambos os lados, o MO1QFA é chamado de *1-way*, já que seu cabeçote de leitura se move em uma única direção,

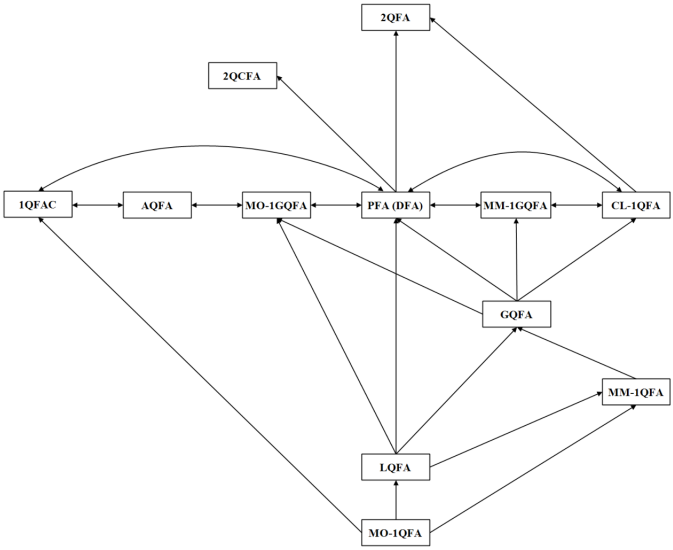

Figura 11 – Relação de inclusão entre os Autômatos Finitos Quânticos [15]. No grafo, setas bidirecionais representam igualdade, os dois autômatos reconhecem o mesmo conjunto de linguagens. E setas unidirecionais indicam que o autômato apontado reconhece mais que o conjunto de linguagens que o autômato apontador reconhece.

sempre em direção ao fim da palavra. No início da computação o cabeçote está no primeiro caractere da palavra e, a cada transição, avança um caractere até o fim da palavra.

Definição 3. *Um MO1QFA é uma quíntupla* $M = (Q, \Sigma, \delta, |q_0\rangle, F)$ *onde:*

$Q = \{|q_0\rangle, \ldots, |q_n\rangle\}$ *é um conjunto finito de estados - Q é contido por um espaço de Hilbert;*

$\Sigma = \{\sigma_0, \ldots, \sigma_k\}$ *é um conjunto finito do alfabeto de entrada;*

δ *é um conjunto de matrizes unitárias,* U_σ*, que descreve as transições entre estados para cada símbolo* σ *do alfabeto de entrada;*

$|q_0\rangle \in Q$ *e é o estado inicial do autômato;*

$F \subseteq Q$ é o conjunto de estados finais.

A computação de uma palavra w é dada por:

$$f(w) = \|P_{acc} U_w |q_0\rangle\|^2, \tag{3.1}$$

onde U_w é operar do caractere inicial ao final da palavra:

$$U = U_{w_{|w|-1}} \dots U_{w_1} U_{w_0}, \tag{3.2}$$

e P_{acc} é o operador de projeção conjunto de estados finais do autômato:

$$P_{acc} = \sum_{q_i \in F} |q_i\rangle\langle q_i| . \tag{3.3}$$

f(w) representa a probabilidade de M aceitar w.

Seguem agora alguns exemplos mostrando o funcionamento do MO1QFA e comparando-o a como seria tal funcionamento nas versões clássicas. Dentro dos exemplos a seguir para fins didáticos de obter a chance de uma palavra ser aceita, são mostrados os valores do estado do sistema, os valores α e β. Em uma execução normal não teríamos acesso a estes valores, somente as respostas 0 ou 1.

3.2.1 Problema do módulo

Say e *Yakaryılmaz* [56] propuseram que, com MO1QFA podemos resolver problemas que possuem erro limitado utilizando menos estados quando comparado à mesma solução em um autômato finito clássico, e para demonstrar isso estudaram o problema do módulo de um número primo.

O reconhecimento com erro limitado possui 3 categorias [42]:

zero-sided o reconhecimento nunca erra, porém em alguns casos, não consegue nem reconhecer nem rejeitar, e portanto retorna que não sabe;

***one-sided* ou unilateral** o reconhecimento ou possui falsos positivos ou falsos negativos, nunca os dois;

***two-sided* ou bilateral** o reconhecimento possui tanto falsos positivos quanto falsos negativos;

Say e *Yakaryılmaz* [56] apresentaram um MO-1QFA que reconhece a linguagem MOD^p com erro limitado, permitindo falsos positivos. A linguagem estudada é definida de modo similar ao feito na seção 2.1.1:

$$\text{MOD}^p = \{a^{jp} | \text{ j é um inteiro não negativo}\} \tag{3.4}$$

A linguagem é sobre o alfabeto unário $\Sigma = \{a\}$, e reconhece sequências de a's múltiplas de um primo p.

Um DFA para reconhecer MOD^p foi apresentado na seção 2.1.1, tal autômato é exato, não possui erro, e possuiria p estados.

No caso de um MO1QFA por outro lado, podemos nos ater a somente dois estados. Para isso, podemos interpretar todas as possíveis possibilidades de superposição de dois estados como na fig. 12; o objetivo é, a partir do estado inicial $|0\rangle$, dar voltas na esfera a cada p a's, assim sempre que a palavra estiver em um a múltiplo de p o estado do sistema será $|0\rangle$.

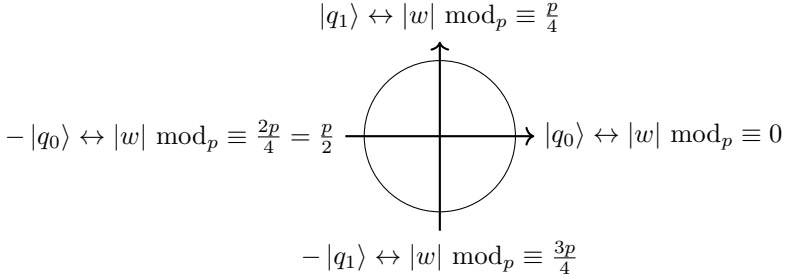

Figura 12 – Conjunto de superposições possíveis com 2 estados. Onde $|w|$ representa o tamanho da palavra w computada.

Para tal, podemos utilizar a seguinte matriz de rotação:

$$A = U_{\theta=\frac{2\pi}{p}} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}, \tag{3.5}$$

onde, com $\theta = \frac{2\pi}{p}$ faremos p transições, saindo de $|0\rangle$ e voltando para $|0\rangle$, porém possibilitando falsos positivos. Nos casos em que o estado do autômato se aproxima de $-|q_0\rangle$ é onde ocorre a maior chance de erro. A chance de um falso positivo é vinculada ao primo em questão e é dada por:

$$\cos^2\left(\frac{\pi}{p}\right) = 1 - \sin^2\left(\frac{\pi}{p}\right). \tag{3.6}$$

Analisando a função de variação do erro com relação ao primo utilizado (fig. 13), podemos ver que quanto maior é o primo p mais próximo de 1 é o erro máximo, ou seja, quanto maior o primo, maior a frequência de falsos positivos e maior é a probabilidade deles serem aceitos. Além disso, é possível ver que o erro associado cresce de maneira de maneira acelerada, atingindo mais de 80% de erro já com primos maiores que 7, o que pode ser um empecilho para casos que não aceitem erro.

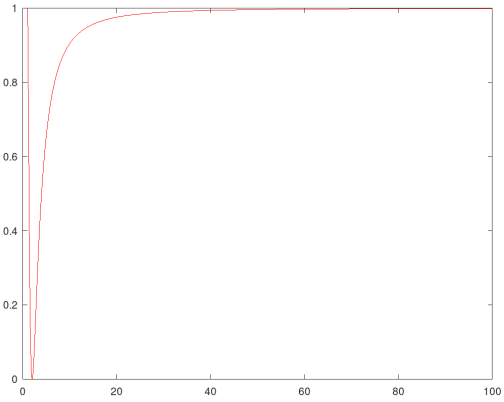

Figura 13 – Variação do erro com relação ao primo utilizado. No eixo x o tamanho do primo p, e no eixo y o valor de erro máximo.

Exemplo - MOD^7

Para MOD^7 a linguagem é definida por:

$$\text{MOD}^7 = \{a^{7j} | j \text{ é um inteiro não negativo}\} \tag{3.7}$$

Um **DFA**, seguindo a notação de Dirac, para MOD^7 pode ser definido por $D = (Q, \Sigma, \delta, q_0, F)$:

Q é $\{|0\rangle,|1\rangle,\ldots,|6\rangle\}$;

Σ é {a};

δ é dado pela seguinte matriz de dimensões 7×7

$$A = \sum_{i=1}^{7} |i\%7\rangle\langle i-1| \tag{3.8}$$

, onde "%"é a operação módulo;

q_0 é $|0\rangle$;

F é $\{|0\rangle\}$.

A computação de $w = aaa$ seria dada por:

$$\begin{aligned} f(w) = \|P_{acc}U_w |q_0\rangle\|^2 &= \||0\rangle\langle 0| AAA |0\rangle\|^2 \\ &= \||0\rangle\langle 0| AA |1\rangle\|^2 = \||0\rangle\langle 0| A |2\rangle\|^2 \\ &= \||0\rangle\langle 0| |3\rangle\|^2 = \|0\|^2 = 0, \end{aligned} \tag{3.9}$$

portanto $w = aaa$ é rejeitada. De fato, podemos verificar que $3\%7 \equiv 3$ e não 0, isso se dá pois, nesta palavra, j não é um inteiro não negativo.

A computação de $w = a^7$ seria dada por:

$$\begin{aligned} f(w) = \|P_{acc}U_w |q_0\rangle\|^2 &= \left\||0\rangle\langle 0| A^7 |0\rangle\right\|^2 \\ &= \left\||0\rangle\langle 0| A^6 |1\rangle\right\|^2 = \left\||0\rangle\langle 0| A^5 |2\rangle\right\|^2 \\ &= \left\||0\rangle\langle 0| A^4 |3\rangle\right\|^2 = \left\||0\rangle\langle 0| A^3 |4\rangle\right\|^2 \\ &= \left\||0\rangle\langle 0| A^2 |5\rangle\right\|^2 = \||0\rangle\langle 0| A |6\rangle\|^2 \\ &= \||0\rangle\langle 0| |0\rangle\|^2 = \||0\rangle\|^2 = \sqrt{1}^2 = 1, \end{aligned} \tag{3.10}$$

portanto $w = a^7$ é aceita como esperado,já que $3\%7 \equiv 0$.

A computação de $w = a^{16}$ seria dada por:

$$\begin{aligned} f(w) = \|P_{acc}U_w\,|q_0\rangle\|^2 &= \left\||0\rangle\langle 0|\,A^{16}\,|0\rangle\right\|^2 \\ = \left\||0\rangle\langle 0|\,A^{15}\,|1\rangle\right\|^2 &= \left\||0\rangle\langle 0|\,A^{14}\,|2\rangle\right\|^2 \\ &\vdots \\ = \left\||0\rangle\langle 0|\,A^{4}\,|5\rangle\right\|^2 &= \left\||0\rangle\langle 0|\,A^{3}\,|6\rangle\right\|^2 \\ = \left\||0\rangle\langle 0|\,A^{2}\,|0\rangle\right\|^2 &= \left\||0\rangle\langle 0|\,A\,|1\rangle\right\|^2 \\ = \left\||0\rangle\langle 0|\,|2\rangle\right\|^2 &= \|0\|^2 = 0, \end{aligned} \tag{3.11}$$

portanto $w = a^{16}$ é rejeitada. De fato, podemos verificar que $16\%7 \equiv 2$ e não 0, isso se dá pois, nesta palavra, j não é um inteiro não negativo.

Por sua vez, um **QFA** para MOD^7 é definido por $M = (Q, \Sigma, \delta, q, F)$:

Q é $\{|0\rangle,|1\rangle\}$ | $|0\rangle$ e $|1\rangle$ são ortonormais;

Σ é $\{a\}$;

δ é dado pela matriz:

$$A = U_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \tag{3.12}$$

onde $\theta = \frac{2\pi}{7}$;

q é $|0\rangle$;

F é $\{|0\rangle\}$.

Outra informação útil nesse caso, é o cálculo da chance de um falso positivo:

$$cos^2\left(\frac{\pi}{p}\right) = cos^2\left(\frac{\pi}{7}\right) \approx 0,81174, \tag{3.13}$$

ou seja, as palavras que possuem a maior taxa de falso positivo terão 81,17% de chance de serem aceitas, mesmo não pertencendo a linguagem.

A computação de $w = aaa$ seria dada por:

$$\begin{aligned} f(w) &= \|P_{acc}U_w\,|q_0\rangle\|^2 = \||0\rangle\langle 0|\,AAA\,|0\rangle\|^2 \\ &= \||0\rangle\langle 0|\,AA(0,62349\,|0\rangle + 0,78183\,|1\rangle)\|^2 \\ &= \||0\rangle\langle 0|\,A(-0,22252\,|0\rangle + 0,97493\,|1\rangle)\|^2 \\ &= \||0\rangle\langle 0|\,(-0,90097\,|0\rangle + 0,43388\,|1\rangle)\|^2 \\ &= \|-0,90097\,|0\rangle\|^2 = \sqrt{(0,81174)^2} = 0,81174, \end{aligned} \tag{3.14}$$

portanto $w = aaa$ possui 81,17% de chance de ser aceita. De fato, podemos verificar que $3\%7 \equiv 3$ e não 0, e portanto deveria ter uma probabilidade de ser aceita menor que 1 e no máximo 81,17%, como calculado anteriormente na eq. (3.13).

A computação de $w = a^7$ seria dada por:

$$\begin{aligned} f(w) &= \|P_{acc}U_w\,|q_0\rangle\|^2 = \big\||0\rangle\langle 0|\,A^7\,|0\rangle\big\|^2 \\ &= \big\||0\rangle\langle 0|\,A^6(0,62349\,|0\rangle + 0,78183\,|1\rangle)\big\|^2 \\ &= \big\||0\rangle\langle 0|\,A^5(-0,22252\,|0\rangle + 0,97493\,|1\rangle)\big\|^2 \\ &= \big\||0\rangle\langle 0|\,A^4(-0,90097\,|0\rangle + 0,43388\,|1\rangle)\big\|^2 \\ &= \big\||0\rangle\langle 0|\,A^3(-0,90097\,|0\rangle - 0,43388\,|1\rangle)\big\|^2 \\ &= \big\||0\rangle\langle 0|\,A^2(-0,22252\,|0\rangle - 0,97493\,|1\rangle)\big\|^2 \\ &= \||0\rangle\langle 0|\,A(0,62349\,|0\rangle - 0,78183\,|1\rangle)\|^2 \\ &= \||0\rangle\langle 0|\,|0\rangle\|^2 = \||0\rangle\|^2 = \sqrt{1}^2 = 1, \end{aligned} \tag{3.15}$$

portanto $w = a^7$ é aceita com 100% de chance como esperado,já que $7\%7 \equiv 0$.

A computação de $w = a^{16}$ seria dada por:

$$\begin{aligned}
f(w) = \|P_{acc}U_w\,|q_0\rangle\|^2 &= \big\||0\rangle\langle 0|\,A^{16}\,|0\rangle\big\|^2 \\
&= \big\||0\rangle\langle 0|\,A^{15}(0,62349\,|0\rangle + 0,78183\,|1\rangle)\big\|^2 \\
&= \big\||0\rangle\langle 0|\,A^{14}(-0,22252\,|0\rangle + 0,97493\,|1\rangle)\big\|^2 \\
&= \big\||0\rangle\langle 0|\,A^{13}(-0,90097\,|0\rangle + 0,43388\,|1\rangle)\big\|^2 \\
&= \big\||0\rangle\langle 0|\,A^{12}(-0,90097\,|0\rangle - 0,43388\,|1\rangle)\big\|^2 \\
&= \big\||0\rangle\langle 0|\,A^{11}(-0,22252\,|0\rangle - 0,97493\,|1\rangle)\big\|^2 \\
&\vdots \\
&= \big\||0\rangle\langle 0|\,A^{3}(0,62349\,|0\rangle - 0,78183\,|1\rangle)\big\|^2 \\
&= \big\||0\rangle\langle 0|\,A^{2}\,|0\rangle\big\|^2 \\
&= \big\||0\rangle\langle 0|\,A(0,62349\,|0\rangle + 0,78183\,|1\rangle)\big\|^2 \\
&= \big\||0\rangle\langle 0|\,(-0,22252\,|0\rangle + 0,97493\,|1\rangle)\big\|^2 \\
= \|-0,22252\,|0\rangle\|^2 &= \sqrt{0,049516}^2 = 0,049516,
\end{aligned} \tag{3.16}$$

portanto $w = a^{16}$ possui 4,95% de probabilidade de ser aceita. De fato, podemos verificar que $16\%7 \equiv 2$ e não 0, e portanto deveria ter uma probabilidade de ser aceita menor que 1 e no máximo 81,17%, como calculado anteriormente na eq. (3.13).

Observação

Foi observado que além de reconhecer MOD^p este autômato também é capaz de reconhecer MOD^n onde n não é um primo ao utilizar $\theta = \frac{\pi}{n}$:

$$A = U_{\theta=\frac{\pi}{n}} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}, \tag{3.17}$$

isso ocorre pois, a alteração do vetor estado ocorre seguindo duas senóides opostas, como na fig. 14.

Olhando a função completa, é possível observar que o período das senoides se repete exatamente na metade do primo selecionado, ou seja, quando

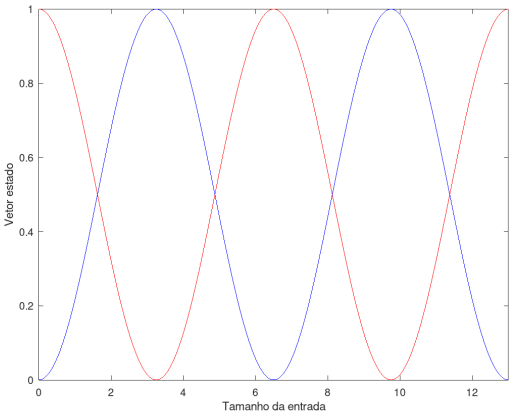

Figura 14 – Variação do vetor estado ao ler 13 caracteres no MO1QFA que reconhece MOD^{13}. Em vermelho se encontra o valor de amplitude do estado $|0\rangle$ e em azul o do $|1\rangle$.

o resultado do módulo p do tamanho da entrada resulta em $\frac{p}{2}$. Assim, ao substituir no ângulo θ o valor do primo p por $2n$ onde n é o natural que se quer o módulo, obtém-se o mapeamento da fig. 15.

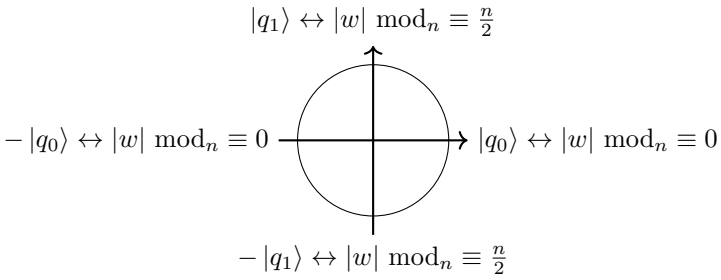

Figura 15 – Conjunto de superposições possíveis com 2 estados. Onde $|w|$ representa o tamanho da palavra w computada.

Este modelo retira os picos observados nos valores congruentes a $\frac{n}{2}$ módulo n, como os existentes na fig. 14 . Além disso, obtém-se a seguinte

matriz de transição:

$$\begin{aligned} \theta &= \frac{2\pi}{p} = \frac{2\pi}{2n} = \frac{\pi}{n}, \\ A = U_{\theta=\frac{\pi}{n}} &= \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}. \end{aligned} \tag{3.18}$$

Além disso, com essa alteração o valor do erro máximo será observado nos valores que tendem a congruência à 0 módulo n, resultando em:

$$cos^2\left((n+1)\frac{\pi}{n}\right), \tag{3.19}$$

que dada a natureza do cosseno, resulta na mesma função definida na eq. (3.6).

3.2.2 Problemas de promessa

Say e *Yakaryılmaz* [56] propuseram que o MO-1QFA poderia resolver problemas de promessa com soluções exatas utilizando menos estados que a solução em um autômato finito.

Um problema de promessa P é definido por uma tupla de duas linguagens $P = (P_s, P_n)$ onde P_s (resp. P_n) é linguagem, consequentemente o conjunto de palavras aceitas (resp. não aceitas). E qualquer outra palavra, as palavras fora de $P_s \cup P_n$, não são importantes, ou seja, podem tanto ser aceitas quanto rejeitadas. Tem-se também que em um problema de promessa $P_s \cap P_n = \varnothing$.

Por sua vez, um problema de promessa é resolvido exatamente por um QFA (resp. DFA) M se: M aceita com probabilidade 1(resp. aceita), toda palavra pertencente a P_s; e M aceita com probabilidade 0 (resp. rejeita), toda palavra pertencente a P_n.

Say e *Yakaryılmaz* [56] estudaram o seguinte problema:

$$P^k = (\text{EVEN}^k, \text{ODD}^k) \tag{3.20}$$

$$\text{EVEN}^k = \{a^{j2^k} \mid \text{j é um inteiro não negativo par}\} \tag{3.21}$$

$$\text{ODD}^k = \{a^{j2^k} \mid \text{j é um inteiro não negativo ímpar}\} \tag{3.22}$$

Um DFA M para resolver P^k precisaria de 2^{k+1} estados [10]. Como, tanto no caso de estar em uma contagem ímpar quanto em uma par, precisamos manter a contagem de quantos dos 2^k a's já foram lidos, o funcionamento poderia ser, inicialmente, separado em duas partes, a par e a ímpar, onde ambas fazem uma contagem de a's. Para resolver módulo de 2^k a's, cada parte precisa contar de 0 até 2^k, e portanto, cada parte precisa de $2^k + 1$ estados.

Ao unir as duas partes podemos reutilizar o estado final da parte par (resp. ímpar) como estado inicial da parte ímpar (resp. par), diminuindo dois estado, totalizando, 2^{k+1} estados. Assim, M difere quando está em uma contagem par ou ímpar, e em ambos os casos, calcula módulo de 2^k. Além disso, o estado inicial e final de M serão o estado inicial da parte par, assim, aceitando somente nos casos em que j é par.

Já um QFA Q pode ser definido de maneira muito mais sucinta, com apenas dois estados. Um estado para representar quando a contagem de 2^k a's é par e um para quando a contagem é ímpar. No estado $|0\rangle$ (resp. $|1\rangle$) temos uma contagem par (resp. ímpar) de 2^k a's até o momento. Podemos analisar o conjunto das possíveis superposições dos estados como um círculo onde cada estado é um eixo (fig. 16). Sabemos que a cada 2^k a's alternamos a contagem de j entre par e ímpar, podemos aplicar essa operação através de rotações de 90°. Com isso, a cada 2^k a's alternamos entre os estados da seguinte forma: $|q_0\rangle$, $|q_1\rangle$, $-|q_0\rangle$, $-|q_1\rangle$ e o ciclo se repete.

Para aplicar uma rotação de $90°\left(\frac{\pi}{2}\right)$ a cada 2^k a's, cada a deve-se rotacionar $\theta = \frac{\pi}{2^{k+1}}$. Como inicialmente não lemos nenhum a, e 0 é uma quantidade par de 2^k a's, o estado inicial será $|q_0\rangle$.

Para aplicar tais rotações temos a seguinte matriz, que será operador de transição para único símbolo do alfabeto:

$$A = U_{\theta=\frac{\pi}{2^{k+1}}} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \tag{3.23}$$

Em uma computação, a cada 4 grupos de 2^k a's, completamos uma

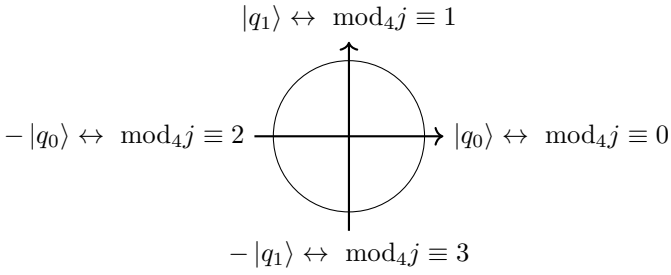

Figura 16 – Conjunto de superposições possíveis com 2 estados.

volta no círculo, a cada 2^k a's alterna-se entre os eixos: $|q_0\rangle$, $|q_1\rangle$, $-|q_0\rangle$, $-|q_1\rangle$. Desta forma, qualquer P^k tem um reconhecimento exato por um QFA de dois estados, uma quantia bem inferior quando comparada aos 2^{k+1} estados na contrapartida clássica (DFA).

Exemplo - P^3

Para P^3 temos:

$$P^3 = (\text{EVEN}^3, \text{ODD}^3)$$
$$\text{EVEN}^3 = \{a^{j2^3} = a^{8j} | \text{ j é um inteiro não negativo par}\} \tag{3.24}$$
$$\text{ODD}^3 = \{a^{j2^3} = a^{8j} | \text{ j é um inteiro não negativo ímpar}\}$$

Um **DFA**, seguindo a notação de Dirac, para P^3 pode ser definido por $D = (Q, \Sigma, \delta, q_0, F)$:

Q é $\{|0\rangle, |1\rangle, \ldots, |15\rangle\}$;

Σ é {a};

δ é dado pela seguinte matriz de dimensões 16×16

$$A = \sum_{i=1}^{16} |i\%16\rangle\langle i-1| \tag{3.25}$$

, onde "%" é a operação módulo;

q_0 é $|0\rangle$;

F é $\{|0\rangle\}$.

A computação de $w = aaa$ seria dada por:

$$\begin{aligned} f(w) = \|P_{acc}U_w\,|q_0\rangle\|^2 &= \||0\rangle\langle 0|\,AAA\,|0\rangle\|^2 \\ &= \||0\rangle\langle 0|\,AA\,|1\rangle\|^2 = \||0\rangle\langle 0|\,A\,|2\rangle\|^2 \\ &= \||0\rangle\langle 0|\,|3\rangle\|^2 = \|0\|^2 = 0 \end{aligned} \tag{3.26}$$

portanto $w = aaa$ é rejeitada.

A computação de $w = a^8$ seria dada por:

$$\begin{aligned} f(w) = \|P_{acc}U_w\,|q_0\rangle\|^2 &= \big\||0\rangle\langle 0|\,A^8\,|0\rangle\big\|^2 \\ &= \big\||0\rangle\langle 0|\,A^7\,|1\rangle\big\|^2 = \big\||0\rangle\langle 0|\,A^6\,|2\rangle\big\|^2 \\ &= \big\||0\rangle\langle 0|\,A^5\,|3\rangle\big\|^2 = \big\||0\rangle\langle 0|\,A^4\,|4\rangle\big\|^2 \\ &= \big\||0\rangle\langle 0|\,A^3\,|5\rangle\big\|^2 = \big\||0\rangle\langle 0|\,A^2\,|6\rangle\big\|^2 \\ &= \||0\rangle\langle 0|\,A\,|7\rangle\|^2 = \||0\rangle\langle 0|\,|8\rangle\|^2 \\ &= \|0\|^2 = 0, \end{aligned} \tag{3.27}$$

portanto $w = a^8$ é rejeitada, o que faz sentido já que para $|w| = 8$ o valor de j em a^{8j} deve ser 1, um número ímpar.

A computação de $w = a^{16}$ seria dada por:

$$\begin{aligned} f(w) = \|P_{acc}U_w\,|q_0\rangle\|^2 &= \big\||0\rangle\langle 0|\,A^{16}\,|0\rangle\big\|^2 \\ &= \big\||0\rangle\langle 0|\,A^{15}\,|1\rangle\big\|^2 = \big\||0\rangle\langle 0|\,A^{14}\,|2\rangle\big\|^2 \\ &= \big\||0\rangle\langle 0|\,A^{13}\,|3\rangle\big\|^2 = \big\||0\rangle\langle 0|\,A^{12}\,|4\rangle\big\|^2 \\ &\vdots \\ &= \big\||0\rangle\langle 0|\,A^3\,|13\rangle\big\|^2 = \big\||0\rangle\langle 0|\,A^2\,|14\rangle\big\|^2 \\ &= \||0\rangle\langle 0|\,A\,|15\rangle\|^2 = \||0\rangle\langle 0|\,|0\rangle\|^2 \\ &= \|1\|^2 = 1, \end{aligned} \tag{3.28}$$

portanto $w = a^{16}$ é aceita, o que faz sentido já que para $|w| = 16$ o valor de j em a^{8j} deve ser 2, um número par.

Por sua vez, um **QFA** para P^3 é definido por $M = (Q, \Sigma, \delta, q, F)$:

Q é $\{|0\rangle,|1\rangle\}$ | $|0\rangle$ e $|1\rangle$ são ortonormais;

Σ é $\{a\}$;

δ é dado pela matriz:

$$A = U_{\theta=\frac{\pi}{2^{3+1}}} = U_{\theta=\frac{\pi}{16}} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}; \tag{3.29}$$

q é $|0\rangle$;

F é $\{|0\rangle\}$.

A computação de $w = aaa$ seria dada por:

$$\begin{aligned} f(w) = \|P_{acc}U_w\,|q_0\rangle\|^2 &= \||0\rangle\langle 0|\,AAA\,|0\rangle\|^2 \\ &= \||0\rangle\langle 0|\,AA(0,98079\,|0\rangle + 0,19509\,|1\rangle)\|^2 \\ &= \||0\rangle\langle 0|\,A(0,92388\,|0\rangle + 0,38268\,|1\rangle)\|^2 \\ &= \||0\rangle\langle 0|\,(0,83147\,|0\rangle + 0,55557\,|1\rangle)\|^2 \\ = \|(0,83147\,|0\rangle)\|^2 &= \sqrt{0,69134}^2 = 0,69134, \end{aligned} \tag{3.30}$$

portanto $w = aaa$ tem $69,13\%$ de chance de ser aceita, lembrando que essa palavra não pertence nem a EVEN^3 nem a ODD^3 então ela ser ou não aceita é irrelevante, o importante são as respostas para os membros de EVEN^3 e ODD^3.

A computação de $w = a^8$ seria dada por:

$$\begin{aligned} f(w) = \|P_{acc}U_w\,|q_0\rangle\|^2 &= \big\||0\rangle\langle 0|\,A^8\,|0\rangle\big\|^2 \\ &= \big\||0\rangle\langle 0|\,A^7(0,98079\,|0\rangle + 0,19509\,|1\rangle)\big\|^2 \\ &= \big\||0\rangle\langle 0|\,A^6(0,92388\,|0\rangle + 0,38268\,|1\rangle)\big\|^2 \\ &= \big\||0\rangle\langle 0|\,A^5(0,83147\,|0\rangle + 0,55557\,|1\rangle)\big\|^2 \\ &= \big\||0\rangle\langle 0|\,A^4(0,70711\,|0\rangle + 0,70711\,|1\rangle)\big\|^2 \\ &= \big\||0\rangle\langle 0|\,A^3(0,55557\,|0\rangle + 0,83147\,|1\rangle)\big\|^2 \\ &= \big\||0\rangle\langle 0|\,A^2(0,38268\,|0\rangle + 0,92388\,|1\rangle)\big\|^2 \\ &= \||0\rangle\langle 0|\,A(0,19509\,|0\rangle + 0,98079\,|1\rangle)\|^2 \\ &= \||0\rangle\langle 0|\,|1\rangle\|^2 = \|0\|^2 = 0, \end{aligned} \tag{3.31}$$

portanto $w = a^8$ tem probabilidade 0 de ser aceita. O que faz bastante sentido já que, neste caso, j é 1, um valor ímpar.

A computação de $w = a^{16}$ seria dada por:

$$\begin{aligned} f(w) = \|P_{acc}U_w\,|q_0\rangle\|^2 &= \big\||0\rangle\langle 0|\,A^{16}\,|0\rangle\big\|^2 \\ &\vdots \\ &= \big\||0\rangle\langle 0|\,A^{12}\,|1\rangle\big\|^2 \\ &\vdots \\ &= \big\||0\rangle\langle 0|\,A^{8}(-|0\rangle)\big\|^2 \\ &\vdots \\ &= \big\||0\rangle\langle 0|\,A^{4}(-|1\rangle)\big\|^2 \\ &\vdots \\ &= \big\||0\rangle\langle 0| - |0\rangle\big\|^2 \\ &= \|-|0\rangle\|^2 = \sqrt{1}^2 = 1, \end{aligned} \tag{3.32}$$

portanto $w = a^{16}$ é aceita. O que faz bastante sentido já que, neste caso, j é 2, um valor par. Interessante ressaltar que, ao tomar a medida, o estado do sistema será $-|0\rangle$, porém essa fase não faz diferença para a medida.

3.2.3 Número de a's diferente do número de b's

Say e *Yakaryılmaz* [56] propuseram que, com MO1QFA não determinístico podemos resolver alguns problemas da classe das linguagens livres de contexto (o conjunto de linguagens reconhecíveis por autômatos de pilha não determinísticos [58]), o que é impossível com autômatos finitos clássicos [58], para isso eles estudaram a linguagem sobre o alfabeto $\Sigma = \{a, b\}$, ou seja, palavras constituídas de a's e b's em qualquer ordem, onde a quantidade de a's é diferente da quantidade de b's. Tal linguagem é definida formalmente da seguinte forma:

$$L = \{(a+b)^* \mid \#a's \neq \#b's\}. \tag{3.33}$$

Um QFA é dito não determinístico quando reconhece com ponto de corte 0 todas as palavras da linguagem, ou seja, com probabilidade maior que 0 reconhece as palavras da linguagem, e com probabilidade igual a 0 reconhece as palavras que não pertencem a linguagem.

Na fig. 9 é mostrado um autômato de Pilha(PFA) com 4 estados que reconhece L.

Por sua vez, é possível definir um MO1QFA para reconhecer essa linguagem com 2 estados de maneira similar aos problemas apresentados anteriormente. Esse autômato tem $|0\rangle$ como estado inicial e $|1\rangle$ como estado final. A ideia neste autômato é utilizarmos do fato que o conjunto de possibilidades formam um círculo, e o fato de que um círculo possui infinitos pontos. Podemos definir rotações no sentido horário para os a's e anti-horário para os b's, assim, durante a computação cada ponto do círculo indica quantos a's ou b's faltam para as quantidades ficarem iguais. Caso o número de a's e b's sejam iguais, o estado do autômato será igual ao estado inicial e, portanto, o resultado da medida será $|0\rangle$ com 100% de chance. Caso contrário, a probabilidade do resultado da medida ser $|1\rangle$ é maior que 0%.

Para executar tais rotações é necessário que o ângulo de rotação seja um múltiplo irracional de π, aqui será utilizado $\sqrt{2}\pi$. Dada a seguinte matriz de rotação, com $\theta = \sqrt{2}\pi$ é feita uma rotação no sentido anti-horário e com $\theta = -\sqrt{2}\pi$ no sentido horário:

$$U_\theta = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}, \tag{3.34}$$

Um MO1QFA para $L = \{(a+b)^* \mid \#a's \neq \#b's\}$ pode ser definido por $M = (Q, \Sigma, \delta, q, F)$:

Q é $\{|0\rangle,|1\rangle\}$ | $|0\rangle$ e $|1\rangle$ são ortonormais;

Σ é $\{a, b\}$;

δ é dado pela matriz:

$$\begin{aligned} A = U_{\theta=-\sqrt{2}\pi} &= \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}, \\ B = U_{\theta=\sqrt{2}\pi} &= \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}; \end{aligned} \tag{3.35}$$

q é $|0\rangle$;

F é $\{|1\rangle\}$.

A computação de $w = aaaba$ seria dada por:

$$\begin{aligned} f(w) = \|P_{acc}U_w\,|q_0\rangle\|^2 &= \||1\rangle\langle 1|\,AAABA\,|0\rangle\|^2 \\ = \||1\rangle\langle 1|\,AAAB(-0,26626\,|0\rangle + 0,96390\,|1\rangle)\|^2 &= \||1\rangle\langle 1|\,AAA\,|0\rangle\|^2 \\ = \||1\rangle\langle 1|\,AA(-0,26626\,|0\rangle + 0,96390\,|1\rangle)\|^2 &= \||1\rangle\langle 1|\,A(-0,85822\,|0\rangle - 0,51329\,|1\rangle)\|^2 \\ = \||1\rangle\langle 1|\,(0,72326\,|0\rangle - 0,69057\,|1\rangle)\|^2 &= \|-0,69057\,|1\rangle\|^2 = \sqrt{0,47689}^2 = 0,47689, \end{aligned} \tag{3.36}$$

portanto $w = aaaba$ tem $47,68\%$ de chance de ser aceita. Já que w possui uma quantidade diferente de a's e b's, a probabilidade de w ser aceita é maior que 0% como o esperado.

A computação de $w = aabb$ seria dada por:

$$\begin{aligned} f(w) = \|P_{acc}U_w\,|q_0\rangle\|^2 &= \||1\rangle\langle 1|\,AABB\,|0\rangle\|^2 \\ = \||1\rangle\langle 1|\,AAB(-0,26626\,|0\rangle - 0,96390\,|1\rangle)\|^2 &= \||1\rangle\langle 1|\,AA(-0,85822\,|0\rangle + 0,51329\,|1\rangle)\|^2 \\ = \||1\rangle\langle 1|\,A(-0,26626\,|0\rangle - 0,96390\,|1\rangle)\|^2 &= \||1\rangle\langle 1|\,|0\rangle\|^2 = \|0\|^2 = 0, \end{aligned} \tag{3.37}$$

portanto $w = aabb$ tem 0% de chance de ser aceita como o esperado, já que w possui uma quantidade igual de a's e b's, e portanto não pertence a linguagem.

A computação de $w = abbaba$ seria dada por:

$$\begin{aligned} f(w) = \|P_{acc}U_w\,|q_0\rangle\|^2 &= \|\,|1\rangle\langle 1|\,ABBABA\,|0\rangle\|^2 \\ &= \|\,|1\rangle\langle 1|\,ABBAB(-0,26626\,|0\rangle + 0,96390\,|1\rangle)\|^2 \\ &= \|\,|1\rangle\langle 1|\,ABBA\,|0\rangle\|^2 \\ &= \|\,|1\rangle\langle 1|\,ABB(-0,26626\,|0\rangle + 0,96390\,|1\rangle)\|^2 \\ &= \|\,|1\rangle\langle 1|\,AB\,|0\rangle\|^2 \\ &= \|\,|1\rangle\langle 1|\,A(-0,26626\,|0\rangle - 0,96390\,|1\rangle)\|^2 \\ &= \|\,|1\rangle\langle 1|\,|0\rangle\|^2 \\ &= \|0\|^2 = 0, \end{aligned} \tag{3.38}$$

portanto $w = abbaba$ tem 0% de chance de ser aceita como o esperado, já que w possui uma quantidade igual de a's e b's, e portanto não pertence a linguagem.

Observação

O autômato apresentado pode ser facilmente alterado para reconhecimento de:

$$L = \{(a+b)^*|\#a's = \#b's\}, \tag{3.39}$$

ou seja, uma linguagem que contém as palavras que possuem quantidades de ocorrências de a's iguais às de b's. A alteração necessária é que o estado inicial e final do autômato seja $|0\rangle$. Porém a alteração proposta possibilita falsos positivos. Isso ocorre pois, quando as quantidades de a's e b's são diferentes, existem somente dois pontos do círculo($\pm\,|1\rangle$) onde não existe chance alguma do estado colapsar em 0.

Feita a apresentação dos trabalhos conhecidos com relação aos autômatos quânticos e uma exemplificação do uso do MO-1QFA, o próximo capítulo realiza experimentos com o MO-1QFA em uma plataforma quântica real e discute seus resultados.

Capítulo 4

Experimentação

Na seção 3.2 foram apresentados algumas aplicações do MO-1QFA's, nesta seção pretende-se estuda-lo de maneira mais prática, a fim de verificar a relevância de QFA nos computadores quânticos disponíveis atualmente. A importância disso, vem do fato de que, além dos sistemas quânticos reais possuírem erros, parte dos autômatos finitos quânticos apresentados também funcionam de maneira probabilística para a maior parte dos problemas. Este capítulo apresentará como mapear um MO-1QFA para o modelo circuital do IBMQ seção 4.1.1, e o estudo de caso do problema do módulo 11, assim como uma discussão sobre sua ocorrência de erros.

4.1 Passos iniciais

A IBM possui o programa IBMQ [30] com o propósito de disseminar o estudo além de incentivar a pesquisa e desenvolvimento de sistemas quânticos, nele estão disponíveis plataformas para simulação e execução de circuitos quânticos, onde códigos quânticos podem ser testados tanto em um simulador, quanto em computadores quânticos reais que a IBM disponibiliza para acesso remoto. As plataformas podem ser utilizados de duas maneiras, através do IBMQ Experience ou através do QisKit.

O IBMQ Experience [29] é uma plataforma online para o desenvolvi-

mento e experimentação de circuitos quânticos, que conta com uma interface gráfica do tipo *drag-and-drop* e um campo para codificar o circuito desejado.

O segundo método é utilizando a biblioteca de código aberto QisKit [1]. Essa API permite a criação de circuitos quânticos através da linguagem python, com experimentação nas diferentes plataformas disponibilizadas por acesso remoto pela IBM.

A experimentação feita neste trabalho utiliza o QisKit e a plataforma ibmq_vigo de 5 qubits.

4.1.1 Mapeamento do MO1QFA para o modelo circuital

O sistema do IBMQ, tanto no QisKit, quanto no IBMQ Experience, utilizam o modelo circuital [19] de computação quântica, isto é, toda a computação é descrita por uma sequência de portas quânticas, onde cada porta representa uma matriz de rotação no qubit especificado.

Todos os modelos apresentados na seção 3.2 possuem somente dois estados, portanto todos eles utilizam 1 qubit, onde o $|0\rangle$ representa um dos estados e o $|1\rangle$ o outro; e cada transição dos autômatos devem ser mapeadas para uma porta lógica.

Para descrever a computação de uma determinada entrada em um determinado autômato no QisKit, devemos descrever a sequência de transições tomadas pelo autômato, ou seja, a sequência de rotações executadas.

O QisKit nos possibilita realizar tais rotações por mais de uma porta, aqui será utilizada a porta $Ry(\theta)$ [2], que é uma porta de 1 qubit, que permite realizar uma rotação do ângulo θ ao redor do eixo y. Podemos utilizá-la para realizar todas as transições nos autômatos descritos na seção 3.2, com um único adendo, pela forma como as portas de um qubit funcionam no QisKit [20] e a matriz utilizada para realizar as rotações da seção 3.2 o ângulo θ especificado no QisKit deve ser o dobro do ângulo que realmente queremos de acordo com a descrição do autômato. Isto se deve

pois a porta $Ry(\theta)$ é mapeada para seguinte matriz [20]:

$$\begin{bmatrix} cos(\frac{\theta_1}{2}) & -sin(\frac{\theta_1}{2}) \\ sin(\frac{\theta_1}{2}) & cos(\frac{\theta_1}{2}). \end{bmatrix}, \tag{4.1}$$

Ao iguala-la com a matriz especificada na seção 3.2 temoS:

$$\begin{bmatrix} cos(\frac{\theta_1}{2}) & -sin(\frac{\theta_1}{2}) \\ sin(\frac{\theta_1}{2}) & cos(\frac{\theta_1}{2}) \end{bmatrix} = \begin{bmatrix} cos(\theta_2) & -sin(\theta_2) \\ sin(\theta_2) & cos(\theta_2) \end{bmatrix}, \tag{4.2}$$

obtemos que $\theta_1 = 2\theta_2$.

Dessa forma o mapeamento do MO1QFA para o modelo circuital do QisKit se dá por uma sequência de portas Ry que descrevem as rotações aplicadas pelas transições tomadas para execução de uma entrada no autômato. Mais precisamente serão utilizadas portas $Ry(2\theta_2)$, onde θ_2 é o ângulo especificado na seção 3.2.

4.2 Problema do módulo 11

Na seção 3.2.1 é mostrado de forma genérica como definir um MO1QFA para resolver o problema do módulo para um primo p:

M é dado pela quíntupla $(Q, \Sigma, \delta, q, F)$:

Q é $\{|0\rangle, |1\rangle\}$ | $|0\rangle$ e $|1\rangle$ são ortonormais;

Σ é $\{a\}$;

δ é dado pela matriz:

$$A = \begin{bmatrix} \cos\frac{2\pi}{p} & -\sin\frac{2\pi}{p} \\ \sin\frac{2\pi}{p} & \cos\frac{2\pi}{p} \end{bmatrix}; \tag{4.3}$$

q é $|0\rangle$;

F é $\{|0\rangle\}$.

De forma mais ilustrativa tem-se o diagrama da fig. 17.

Neste caso de estudo, será explorado o erro que ocorre durante a execução do MO1QFA que resolve módulo 11 sendo executado pelo QisKit.

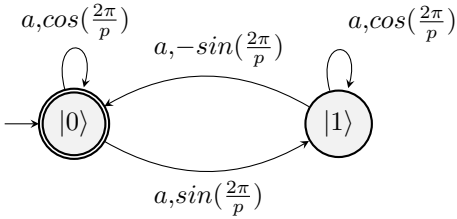

Figura 17 – Autômato que reconhece uma palavra com número de a's múltiplo de p.

4.2.1 Autômato e mapeamento

Um MO1QFA que resolve módulo para o primo 11 é definido pela quíntupla $(Q, \Sigma, \delta, q, F)$:

Q é $\{|0\rangle,|1\rangle\}$ | $|0\rangle$ e $|1\rangle$ são ortonormais;

Σ é $\{a\}$;

δ é dado pela matriz:

$$A = \begin{bmatrix} cos\frac{2\pi}{11} & -sin\frac{2\pi}{11} \\ sin\frac{2\pi}{11} & cos\frac{2\pi}{11} \end{bmatrix}; \tag{4.4}$$

q é $|0\rangle$;

F é $\{|0\rangle\}$.

E pode ser visualizado pelo seguinte diagrama de estados:

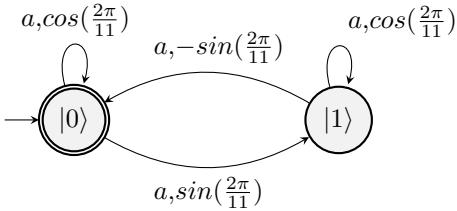

Figura 18 – Autômato que reconhece uma palavra com número de a's múltiplo de 11.

Como o ângulo em A é $\frac{2\pi}{11}$ ao mapear para a porta Ry do QisKit será utilizado o seguinte valor θ:

$$\theta = 2 \times \frac{2\pi}{11} = \frac{4\pi}{11}. \tag{4.5}$$

Portanto, os testes feitos tiveram a quantidade de portas $Ry(\frac{4\pi}{11})$ igual ao números de a's da entrada, por exemplo, se a entrada tiver k a's, serão incluídos k portas $Ry(\frac{4\pi}{11})$ e em seguida a porta de medida. Mais informações sobre a implementação e o uso do QisKit podem ser encontradas no apêndice A.

A probabilidade teórica do autômato aceitar uma dada entrada w é dada por:

$$\begin{aligned} f(w) &= \|P_{acc} U_w q_0\|^2 \\ &= \left\| |0\rangle\langle 0| \, A^{|w|} \, |0\rangle \right\|^2. \end{aligned} \tag{4.6}$$

Dado a natureza de A, $A^{|w|}$ é:

$$A^{|w|} = \begin{bmatrix} \cos\left(\frac{2\pi}{11}|w|\right) & -\sin\left(\frac{2\pi}{11}|w|\right) \\ \sin\left(\frac{2\pi}{11}|w|\right) & \cos\left(\frac{2\pi}{11}|w|\right) \end{bmatrix}, \tag{4.7}$$

portanto:

$$\begin{aligned} f(w) &= \left\| |0\rangle\langle 0| \, A^{|w|} \, |0\rangle \right\|^2 \\ &= \left\| \left[cos\left(\frac{2\pi}{11}|w|\right) \quad -sin\left(\frac{2\pi}{11}|w|\right) \right] |0\rangle \right\|^2 \\ &= \left\| cos\left(\frac{2\pi}{11}|w|\right) \right\|^2 \\ &= cos^2\left(\frac{2\pi}{11}|w|\right), \end{aligned} \tag{4.8}$$

onde $|w|$ é o tamanho da palavra w.

4.2.2 Casos estudados, resultados e discussão

Inicialmente os casos de teste estudados foram as instâncias com tamanho entre 0 e 1000 que são congruentes a 0 ou a 3 módulo 11, ou seja,

foram testados 182 entradas diferentes, cada uma com 8192 execuções no ibmq_vigo.

Teoricamente(eq. (4.8)), é esperado que os valores congruentes a 0 módulo 11 tivessem 100% de probabilidade de serem aceitos e os valores congruentes a 3 módulo 11 tivessem 2,0253% de probabilidade de serem aceitos.

Nas figuras (fig. 19 a fig. 21) podemos observar e comparar os valores reais com os obtidos. Os valores do erro absoluto(módulo da diferença entre a probabilidade esperada e a probabilidade real obtida nos experimentos) obtidos podem ser visualizados nas tabelas (tabela 3 a tabela 5).

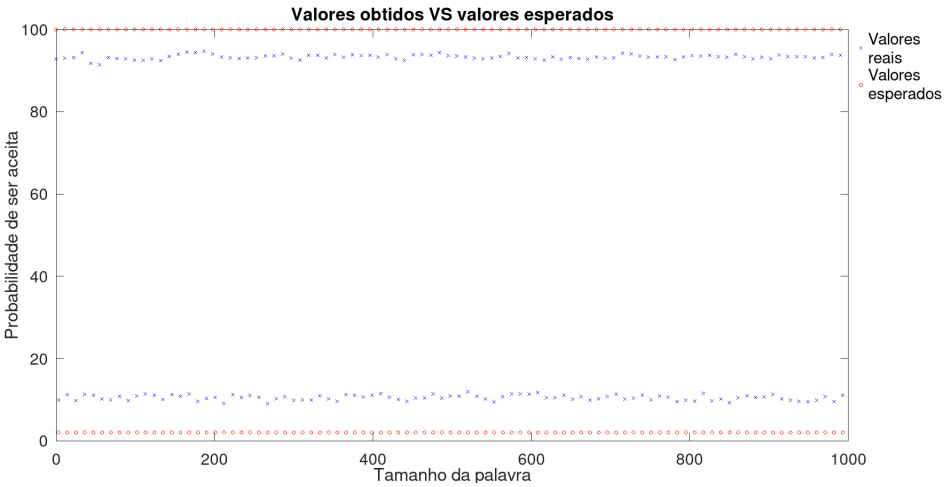

Figura 19 – Comparação das probabilidades esperadas, em vermelho, e a probabilidade obtida, em azul. Os valores de cima são referentes aos tamanhos congruentes a 0 módulo 11, e os de baixo aos tamanhos congruentes a 3 módulo 11.

Na fig. 19 e tabela 3 podemos ver todos os casos analisados, o valor de erro é alto (com uma média de aproximadamente 7.579 pontos) e ocorre de maneira uniforme durante todo o intervalo (com um desvio padrão de aproximadamente 1.98), ou seja, parece ignorar a decoerência que deveria ocorrer na aplicação de uma quantidade alta de portas, como é o caso de

Erro máximo	9.900918211974883	Tamanho da entrada	520
Erro mínimo	5.2978515625	Tamanho da entrada	187
Média		7.578928049153639	
Variância		1.205517567315090	
Desvio padrão		1.097960640148403	

Tabela 3 – Resumo dos valores de erro absoluto obtidos.

aplicar centenas de portas.

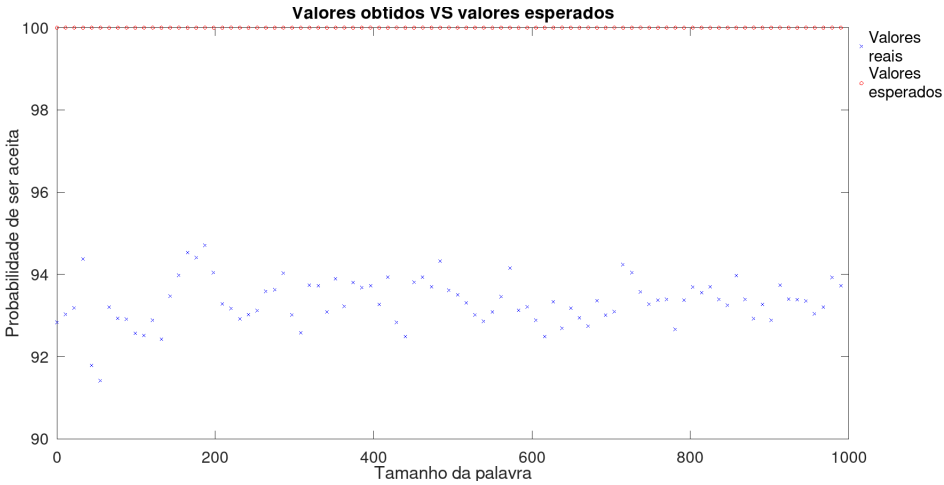

Figura 20 – Comparação das probabilidades esperadas, em vermelho, e a probabilidade obtida, em azul. Os valores são referentes ás entradas com tamanho congruente a 0 módulo 11.

Erro máximo	8.58154296875	Tamanho da entrada	55
Erro mínimo	5.2978515625	Tamanho da entrada	187
Média		6.664502489697802	
Variância		0.3123444256150856	
Desvio padrão		0.5588778270920091	

Tabela 4 – Resumo dos valores de erro absoluto obtidos para entradas com tamanho congruente a 0 módulo 11.

Na fig. 20 e tabela 4 podemos ver os casos congruentes a 0. Aqui o erro ainda é alto, porém ligeiramente menor (com uma média de aproximada-

mente 6.665 pontos, quase um ponto a menos que a média geral) e ocorre de maneira uniforme (com um desvio padrão de aproximadamente 0.559).

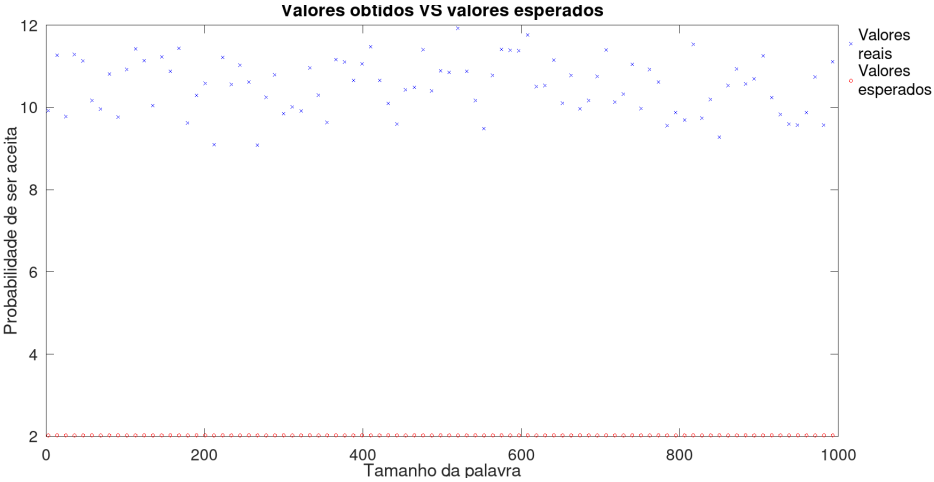

Figura 21 – Comparação das probabilidades esperadas, em vermelho, e a probabilidade obtida, em azul. Os valores são referentes às entradas com tamanho congruente a 3 módulo 11.

Erro máximo	9.900918211974883	Tamanho da entrada	520
Erro mínimo	7.056679930724883	Tamanho da entrada	267
Média		8.493353608609491	
Variância		0.4263425014428133	
Desvio padrão		0.6529490802833046	

Tabela 5 – Resumo dos valores de erro absoluto obtidos para entradas com tamanho congruente a 3 módulo 11.

Na fig. 21 e tabela 5 podemos ver os casos congruentes a 3. Aqui o erro é mais alto que no caso dos congruentes a 0 (com uma média de aproximadamente 8.49 pontos, quase um ponto a mais que a média geral e quase dois pontos a mais que no caso dos congruentes a 0) e ocorre de maneira uniforme (com um desvio padrão de aproximadamente 0.653, ligeiramente maior que no caso dos congruentes a 0).

Para entender os fatores que compõem o erro que ocorrem durante a

computação foi definido uma função para aproximar os valores obtidos do valor esperado, dessa forma talvez se possa entender quais erros estão ocorrendo, e qual o impacto de cada um.

O erro pode ser definido por 3 fatores principais: α (representa um erro de amplitude), δ (representa um erro de fase) e β (um ruído).

Com a aplicação dos erros a computação seria alterada. Sairia da computação originalmente definida por:

$$\cos^2\left(\frac{2\pi}{11}|w|\right), \tag{4.9}$$

para a influênciada pelo erro:

$$\alpha\cos^2\left(\frac{2\pi}{11}|w|+\delta\right)+\beta. \tag{4.10}$$

A função de aproximação definida, focou nos valores de α e β, já que, nesse caso, erros de fase não seriam relevantes. Foram obtidos, com testes empíricos, os valores de 0.85 e 8.5 referentes respectivamente a α e β:

$$0.85\times cos^2\left(\frac{2\pi}{11}|w|\right)+8.5, \tag{4.11}$$

O erro entre os valores obtidos e a função aproximada é pequeno, como apresentado na tabela 6.

Média	0.5329689393703508
Variância	0.1438948867048030
Desvio padrão	0.3793347950093729

Tabela 6 – Resumo dos valores de erro obtidos após a função de aproximação.

O que indica que os valores α e β obtidos são próximos aos reais valores de erro.

Percebe-se que valores melhores para α e β podem ser obtidos a partir dos erros já apresentados nas tabela 3 a tabela 5. β pode ser definido como o erro médio dos valores com tamanho congruentes a 3 módulo 11, o

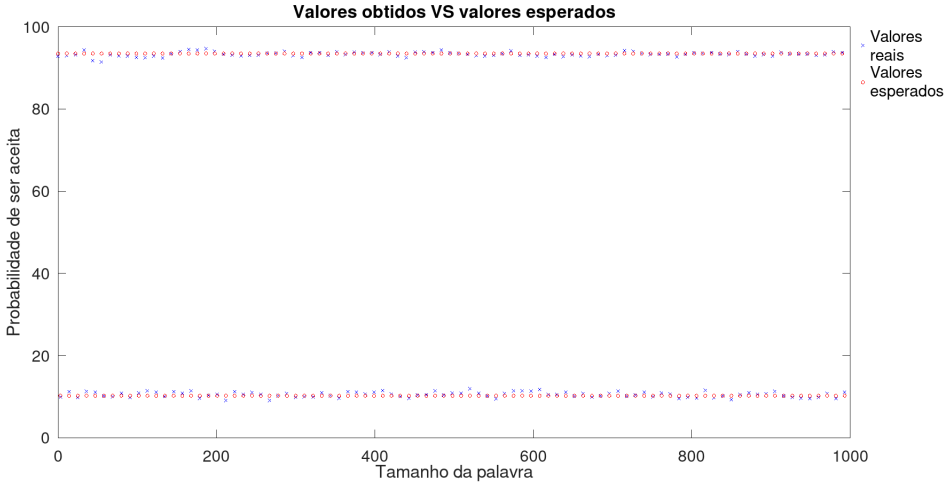

Figura 22 – Comparação das probabilidades esperada após a função de aproximação, em vermelho, e a probabilidade obtida, em azul.

que "levanta" a função ao ser somado. E α pode ser definido pela seguinte equação:

$$\alpha = (100 - E_0 - \beta)/100 \approx 0.848, \tag{4.12}$$

onde E_0 representa o erro médio dos valores com tamanho congruente 0 módulo 11.

Como os valores de ótimos são bem próximos dos valores obtidos empiricamente, o erro médio, variância e desvio padrão não são muito alterados, com diferenças somente a partir da segunda casa decimal(tabela 7).

Média	0.51903759262535
Variância	0.1470678913781397
Desvio padrão	0.3834943172696822

Tabela 7 – Resumo dos valores de erro obtidos após a função de aproximação com valores α e β ótimos.

Capítulo 5

Conclusão

5.1 Considerações Finais

O presente trabalho apresentou um total de 16 modelos de autômatos finitos e de pilha quânticos e algumas relações conhecidas entre eles e entre os modelos clássicos. Concentrou-se no MO-1QFA para demonstrar problemas tratáveis com o modelo mais simples de QFA, e demonstrar diferenças de implementação entre AF's e QFA's. Entre os exemplos, demonstrou que os QFA's resolvem algumas classes de problemas utilizando menos memória, embora para isso haja um sacrifício na certeza da resposta. E que, além disso, os QFA's extrapolam a classe das Linguagens Regulares, mostrando que eles podem realizar tarefas que AF's não podem. O presente trabalho também trouxe um estudo sobre a implementação de um problema resolvido por meio de um autômato finito quântico em um computador quântico real, e discutiu sobre as capacidades atuais de implementação deste modelo.

5.2 Trabalhos Futuros

É importante estudar outros modelos, além do MO-1QFA, para se ter um entendimento sobre como podem ser utilizados, de forma a obtermos

um maior detalhamento sobre como desenvolvê-os para resolver uma dada linguagem. Em especial os autômatos de pilha. Outro assunto a se abordar é com relação aos testes na plataforma da IBM, é importante um estudo mais aprofundado sobre os causadores dos erros, e de testes em outras plataformas para comparação dos resultados. Seria interessante também a implementação de mais autômatos e o estudo de seu mapeamento para o modelo circuital de computação. Espera-se que com o estudo desse erro, possa se estudar sua mitigação. Espera-se que esse trabalho sirva de inspiração para mais estudos com relação a versões quânticas de autômatos finitos e de pilha, além do estudo das suas classes de complexidade.

Referências

[1] Héctor Abraham, Ismail Yunus Akhalwaya, Gadi Aleksandrowicz, Thomas Alexander, Gadi Alexandrowics, Eli Arbel, Abraham Asfaw, Carlos Azaustre, Panagiotis Barkoutsos, George Barron, Luciano Bello, Yael Ben-Haim, Daniel Bevenius, Lev S. Bishop, Samuel Bosch, David Bucher, CZ, Fran Cabrera, Padraic Calpin, Lauren Capelluto, Jorge Carballo, Ginés Carrascal, Adrian Chen, Chun-Fu Chen, Richard Chen, Jerry M. Chow, Christian Claus, Christian Clauss, Abigail J. Cross, Andrew W. Cross, Juan Cruz-Benito, Cryoris, Chris Culver, Antonio D. Córcoles-Gonzales, Sean Dague, Matthieu Dartiailh, Abdón Rodríguez Davila, Delton Ding, Eugene Dumitrescu, Karel Dumon, Ivan Duran, Pieter Eendebak, Daniel Egger, Mark Everitt, Paco Martín Fernández, Albert Frisch, Andreas Fuhrer, IAN GOULD, Julien Gacon, Gadi, Borja Godoy Gago, Jay M. Gambetta, Luis Garcia, Shelly Garion, Gawel-Kus, Juan Gomez-Mosquera, Salvador de la Puente González, Donny Greenberg, John A. Gunnels, Isabel Haide, Ikko Hamamura, Vojtech Havlicek, Joe Hellmers, Łukasz Herok, Hiroshi Horii, Connor Howington, Shaohan Hu, Wei Hu, Haruki Imai, Takashi Imamichi, Raban Iten, Toshinari Itoko, Ali Javadi-Abhari, Jessica, Kiran Johns, Naoki Kanazawa, Anton Karazeev, Paul Kassebaum, Arseny Kovyrshin, Vivek Krishnan, Kevin Krsu-

lich, Gawel Kus, Ryan LaRose, Raphaël Lambert, Joe Latone, Scott Lawrence, Dennis Liu, Peng Liu, Panagiotis Barkoutsos ZRL Mac, Yunho Maeng, Aleksei Malyshev, Jakub Marecek, Manoel Marques, Dolph Mathews, Atsushi Matsuo, Douglas T. McClure, Cameron McGarry, David McKay, Srujan Meesala, Antonio Mezzacapo, Rohit Midha, Zlatko Minev, Michael Duane Mooring, Renier Morales, Niall Moran, Prakash Murali, Jan Müggenburg, David Nadlinger, Giacomo Nannicini, Paul Nation, Yehuda Naveh, Nick-Singstock, Pradeep Niroula, Hassi Norlen, Lee James O'Riordan, Pauline Ollitrault, Steven Oud, Dan Padilha, Hanhee Paik, Simone Perriello, Anna Phan, Marco Pistoia, Alejandro Pozas-iKerstjens, Viktor Prutyanov, Jesús Pérez, Quintiii, Rudy Raymond, Rafael Martín-Cuevas Redondo, Max Reuter, Diego M. Rodríguez, Mingi Ryu, Martin Sandberg, Ninad Sathaye, Bruno Schmitt, Chris Schnabel, Travis L. Scholten, Eddie Schoute, Ismael Faro Sertage, Nathan Shammah, Yunong Shi, Adenilton Silva, Yukio Siraichi, Seyon Sivarajah, John A. Smolin, Mathias Soeken, Dominik Steenken, Matt Stypulkoski, Hitomi Takahashi, Charles Taylor, Pete Taylour, Soolu Thomas, Mathieu Tillet, Maddy Tod, Enrique de la Torre, Kenso Trabing, Matthew Treinish, TrishaPe, Wes Turner, Yotam Vaknin, Carmen Recio Valcarce, Francois Varchon, Desiree Vogt-Lee, Christophe Vuillot, James Weaver, Rafal Wieczorek, Jonathan A. Wildstrom, Robert Wille, Erick Winston, Jack J. Woehr, Stefan Woerner, Ryan Woo, Christopher J. Wood, Ryan Wood, Stephen Wood, James Wootton, Daniyar Yeralin, Jessie Yu, Laura Zdanski, Zoufalc, anedumla, azulehner, bcamorrison, brandhsn, dennis-liu 1, drholmie, elfrocampeador, fanizzamarco, gruu, kanejess, klinvill, lerongil, ma5x, merav aharoni, mrossinek, ordmoj, strickroman, tigerjack, yang.luh, and yotamvakninibm. Qiskit: An open-source framework for quantum computing, 2019.

[2] Héctor Abraham, Ismail Yunus Akhalwaya, Gadi Aleksandrowicz, Thomas Alexander, Gadi Alexandrowics, Eli Arbel, Abraham Asfaw,

Carlos Azaustre, Panagiotis Barkoutsos, George Barron, Luciano Bello, Yael Ben-Haim, Daniel Bevenius, Lev S. Bishop, Samuel Bosch, David Bucher, CZ, Fran Cabrera, Padraic Calpin, Lauren Capelluto, Jorge Carballo, Ginés Carrascal, Adrian Chen, Chun-Fu Chen, Richard Chen, Jerry M. Chow, Christian Claus, Christian Clauss, Abigail J. Cross, Andrew W. Cross, Juan Cruz-Benito, Cryoris, Chris Culver, Antonio D. Córcoles-Gonzales, Sean Dague, Matthieu Dartiailh, Abdón Rodríguez Davila, Delton Ding, Eugene Dumitrescu, Karel Dumon, Ivan Duran, Pieter Eendebak, Daniel Egger, Mark Everitt, Paco Martín Fernández, Albert Frisch, Andreas Fuhrer, IAN GOULD, Julien Gacon, Gadi, Borja Godoy Gago, Jay M. Gambetta, Luis Garcia, Shelly Garion, Gawel-Kus, Juan Gomez-Mosquera, Salvador de la Puente González, Donny Greenberg, John A. Gunnels, Isabel Haide, Ikko Hamamura, Vojtech Havlicek, Joe Hellmers, Łukasz Herok, Hiroshi Horii, Connor Howington, Shaohan Hu, Wei Hu, Haruki Imai, Takashi Imamichi, Raban Iten, Toshinari Itoko, Ali Javadi-Abhari, Jessica, Kiran Johns, Naoki Kanazawa, Anton Karazeev, Paul Kassebaum, Arseny Kovyrshin, Vivek Krishnan, Kevin Krsulich, Gawel Kus, Ryan LaRose, Raphaël Lambert, Joe Latone, Scott Lawrence, Dennis Liu, Peng Liu, Panagiotis Barkoutsos ZRL Mac, Yunho Maeng, Aleksei Malyshev, Jakub Marecek, Manoel Marques, Dolph Mathews, Atsushi Matsuo, Douglas T. McClure, Cameron McGarry, David McKay, Srujan Meesala, Antonio Mezzacapo, Rohit Midha, Zlatko Minev, Michael Duane Mooring, Renier Morales, Niall Moran, Prakash Murali, Jan Müggenburg, David Nadlinger, Giacomo Nannicini, Paul Nation, Yehuda Naveh, Nick-Singstock, Pradeep Niroula, Hassi Norlen, Lee James O'Riordan, Pauline Ollitrault, Steven Oud, Dan Padilha, Hanhee Paik, Simone Perriello, Anna Phan, Marco Pistoia, Alejandro Pozas-iKerstjens, Viktor Prutyanov, Jesús Pérez, Quintiii, Rudy Raymond, Rafael Martín-Cuevas Redondo, Max Reuter, Diego M. Rodríguez, Mingi

Ryu, Martin Sandberg, Ninad Sathaye, Bruno Schmitt, Chris Schnabel, Travis L. Scholten, Eddie Schoute, Ismael Faro Sertage, Nathan Shammah, Yunong Shi, Adenilton Silva, Yukio Siraichi, Seyon Sivarajah, John A. Smolin, Mathias Soeken, Dominik Steenken, Matt Stypulkoski, Hitomi Takahashi, Charles Taylor, Pete Taylour, Soolu Thomas, Mathieu Tillet, Maddy Tod, Enrique de la Torre, Kenso Trabing, Matthew Treinish, TrishaPe, Wes Turner, Yotam Vaknin, Carmen Recio Valcarce, Francois Varchon, Desiree Vogt-Lee, Christophe Vuillot, James Weaver, Rafal Wieczorek, Jonathan A. Wildstrom, Robert Wille, Erick Winston, Jack J. Woehr, Stefan Woerner, Ryan Woo, Christopher J. Wood, Ryan Wood, Stephen Wood, James Wootton, Daniyar Yeralin, Jessie Yu, Laura Zdanski, Zoufalc, anedumla, azulehner, bcamorrison, brandhsn, dennis-liu 1, drholmie, elfrocampeador, fanizzamarco, gruu, kanejess, klinvill, lerongil, ma5x, merav aharoni, mrossinek, ordmoj, strickroman, tigerjack, yang.luh, and yotamvakninibm. Qiskit - class ry gate. Disponivel em: `https://qiskit.org/documentation/api/qiskit.extensions.RYGate.html?highlight=ry%20gate#qiskit.extensions.RYGate`. Acessado em 2019-10-22.

[3] Masami Amano and Kazuo Iwama. Undecidability on quantum finite automata. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 368–375. ACM, 1999.

[4] Andris Ambainis, Martin Beaudry, Marats Golovkins, Arnolds Ķikusts, Mark Mercer, and Denis Thérien. Algebraic results on quantum automata. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 93–104. Springer, 2004.

[5] Andris Ambainis and Rusins Freivalds. 1-way quantum finite automata: strengths, weaknesses and generalizations. In *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No. 98CB36280)*, pages 332–341. IEEE, 1998.

[6] Andris Ambainis and Arnolds Ķikusts. Exact results for accepting probabilities of quantum automata. *Theoretical Computer Science*, 295(1-3):3–25, 2003.

[7] Andris Ambainis, Arnolds KĶikusts, and Māris Valdats. On the class of languages recognizable by 1-way quantum finite automata. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 75–86. Springer, 2001.

[8] Andris Ambainis, Ashwin Nayak, Amnon Ta-Shma, and Umesh Vazirani. Dense quantum coding and a lower bound for 1-way quantum automata. *arXiv preprint quant-ph/9804043*, 1998.

[9] Andris Ambainis and John Watrous. Two-way finite automata with quantum and classical states. *Theoretical Computer Science*, 287(1):299–311, 2002.

[10] Andris Ambainis and Abuzer Yakaryılmaz. Superiority of exact quantum automata for promise problems. *Information Processing Letters*, 112(7):289–291, 2012.

[11] Andris Ambainis and Abuzer Yakaryılmaz. Automata and quantum computing. *arXiv preprint arXiv:1507.01988*, 2015.

[12] Gene M Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference*, pages 483–485. ACM, 1967.

[13] Alberto Bertoni and Marco Carpentieri. Analogies and differences between quantum and stochastic automata. *Theoretical Computer Science*, 262(1-2):69–81, 2001.

[14] Alberto Bertoni, Carlo Mereghetti, and Beatrice Palano. Quantum computing: 1-way quantum automata. In *International Conference on Developments in Language Theory*, pages 1–20. Springer, 2003.

[15] Amandeep Singh Bhatia and Ajay Kumar. Quantum finite automata: survey, status and research directions. *arXiv preprint arXiv:1901.07992*, 2019.

[16] Alex Brodsky and Nicholas Pippenger. Characterizations of 1-way quantum finite automata. *SIAM Journal on Computing*, 31(5):1456–1478, 2002.

[17] Robert H Dennard, Fritz H Gaensslen, V Leo Rideout, Ernest Bassous, and Andre R LeBlanc. Design of ion-implanted mosfet's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, 9(5):256–268, 1974.

[18] David Deutsch. Quantum theory, the church–turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 400(1818):97–117, 1985.

[19] IBM Q Experience. Introduction to quantum circuits - getting started with quantum circuits. Disponivel em `https://quantum-computing.ibm.com/support/guides/introduction-to-quantum-circuits?section=5cae613866c1694be21df8cc`. Acessado em: 2019-10-22.

[20] IBM Q Experience. Introduction to quantum circuits - quantum gates. Disponivel em: `https://quantum-computing.ibm.com/support/guides/introduction-to-quantum-circuits?page=5cae6f7735dafb4c01214bbe#other-single-qubit-gates`. Acesso em: 22/10/2019.

[21] Richard P Feynman. Simulating physics with computers. *International journal of theoretical physics*, 21(6):467–488, 1982.

[22] Max H Garzon and Russell J Deaton. Biomolecular computing and programming. *IEEE Transactions on Evolutionary Computation*, 3(3):236–250, 1999.

[23] Marats Golovkins. Quantum pushdown automata. In *International Conference on Current Trends in Theory and Practice of Computer Science*, pages 336–346. Springer, 2000.

[24] Lov K Grover. A fast quantum mechanical algorithm for database search. *arXiv preprint quant-ph/9605043*, 1996.

[25] Stanley Gudder. Quantum computers. *International Journal of Theoretical Physics*, 39(9):2151–2177, 2000.

[26] John L Hennessy and David A Patterson. *Computer architecture: a quantitative approach*. Elsevier, 2018.

[27] Mika Hirvensalo. Quantum automata with open time evolution. *International Journal of Natural Computing Research (IJNCR)*, 1(1):70–85, 2010.

[28] John E Hopcroft. *Introduction to automata theory, languages, and computation*. Pearson Education India, 2008.

[29] IBM. Ibm q experience. Disponivel em : `https://www.ibm.com/quantum-computing/technology/experience/`. Acessado em: 2019-10-22.

[30] IBM. Ibm quantum computing. Disponivel em: `https://www.ibm.com/quantum-computing/`. Acessado em: 2019-10-22.

[31] Arnolds Kikusts. A small 1-way quantum finite automaton. *arXiv preprint quant-ph/9810065*, 1998.

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

[33] Takeshi Koshiba. Polynomial-time algorithms for the equivalence for one-way quantum finite automata. In *International Symposium on Algorithms and Computation*, pages 268–278. Springer, 2001.

[34] Lvzhou Li and Daowen Qiu. A polynomial-time algorithm for the equivalence between quantum sequential machines. *arXiv preprint quant-ph/0604085*, 2006.

[35] Lvzhou Li and Daowen Qiu. Determining the equivalence for one-way quantum finite automata. *Theoretical Computer Science*, 403(1):42–51, 2008.

[36] Lvzhou Li, Daowen Qiu, Xiangfu Zou, Lvjun Li, Lihua Wu, and Paulo Mateus. Characterizations of one-way general quantum finite automata. *Theoretical Computer Science*, 419:73–91, 2012.

[37] Yu-Ming Lin, Alberto Valdes-Garcia, Shu-Jen Han, Damon B Farmer, Inanc Meric, Yanning Sun, Yanqing Wu, Christos Dimitrakopoulos, Alfred Grill, Phaedon Avouris, et al. Wafer-scale graphene integrated circuit. *Science*, 332(6035):1294–1297, 2011.

[38] Victor S Miller. Use of elliptic curves in cryptography. In *Conference on the theory and application of cryptographic techniques*, pages 417–426. Springer, 1985.

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

[40] Gordon E Moore et al. Cramming more components onto integrated circuits, 1965.

[41] Gordon E Moore et al. Progress in digital integrated electronics. In *Electron Devices Meeting*, volume 21, pages 11–13, 1975.

[42] Nabil Mustafa. Complexity theory - randimized computation. Disponivel em: `https://perso.esiee.fr/~mustafan/TechnicalWritings/Complexityslides/lec14.pdf`. Acessado em: 2019-06-03.

[43] Masaki Nakanishi. On the power of one-sided error quantum pushdown automata with classical stack operations. In *International Computing and Combinatorics Conference*, pages 179–187. Springer, 2004.

[44] Michael A Nielsen and Isaac L Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, 10th anniversary edition edition, 2010.

[45] K Paschen. Quantum finite automata using ancilla qubits. university of karlsruhe. Technical report, Technical report, 2000.

[46] Daowen Qiu. Automata theory based on quantum logic: some characterizations. *Information and Computation*, 190(2):179–195, 2004.

[47] Daowen Qiu. Automata theory based on quantum logic: Reversibilities and pushdown automata. *Theoretical Computer Science*, 386(1-2):38–56, 2007.

[48] Daowen Qiu. Some observations on two-way finite automata with quantum and classical states. In *International Conference on Intelligent Computing*, pages 1–8. Springer, 2008.

[49] Daowen Qiu and Lvzhou Li. An overview of quantum computation models: quantum automata. *Frontiers of Computer Science in China*, 2(2):193–207, 2008.

[50] Daowen Qiu and Mingsheng Ying. Characterizations of quantum automata. *Theoretical computer science*, 312(2-3):479–489, 2004.

[51] DW Qiu, Paulo Mateus, and A Sernadas. One-way quantum finite automata together with classical states. *arXiv preprint arXiv:0909.1428*, pages 3006–3017, 2009.

[52] Michael O Rabin. Probabilistic automata. *Information and control*, 6(3):230–245, 1963.

[53] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[54] Martin Roetteler, Michael Naehrig, Krysta M Svore, and Kristin Lauter. Quantum resource estimates for computing elliptic curve discrete logarithms. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 241–270. Springer, 2017.

[55] Arto Salomaa. Seymour ginsburg. algebraic and automata-theoretic properties of formal languages. fundamental studies in computer science, vol. 2. north-holland publishing company, amsterdam and oxford, and american elsevier publishing company, inc., new york, 1975, xii+ 313 pp. *The Journal of Symbolic Logic*, 41(4):788–789, 1976.

[56] AC Cem Say and Abuzer Yakaryılmaz. Quantum finite automata: A modern introduction. In *Computing with New Resources*, pages 208–222. Springer, 2014.

[57] Peter W Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.

[58] Michael Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 2012.

[59] Jiacun Wang. *Handbook of Finite State Based Models and Applications*. CRC press, 2012.

[60] Abuzer Yakaryilmaz and AC Say. Languages recognized by nondeterministic quantum finite automata. *arXiv preprint arXiv:0902.2081*, 2009.

[61] Abuzer Yakaryilmaz and AC Cem Say. Languages recognized with unbounded error by quantum finite automata. In *International Computer Science Symposium in Russia*, pages 356–367. Springer, 2009.

[62] Abuzer Yakaryılmaz and AC Cem Say. Unbounded-error quantum computation with small space bounds. *Information and Computation*, 209(6):873–892, 2011.

[63] Mingsheng Ying. Automata theory based on quantum logic ii. *International Journal of Theoretical Physics*, 39(11):2545–2557, 2000.

APÊNDICE A

Usando o QisKit

Neste apêndice é apresentado o uso do QisKit para computar as instancias da seção 4.2.2 para o problema do modulo 11 no *ibmq_vigo* e em seguida uma ligeira explicação sobre o código.

```
1  from qiskit import *
2  from qiskit.tools.jupyter import *
3  from qiskit.visualization import *
4  from qiskit.tools.monitor import job_monitor
5  import numpy as np
6
7  # If your account is not logged you can save ir like:
8  #IBMQ.save_account('code')
9  IBMQ.load_account()
10
11 # Here ibmq_vigo is setted as the backend to be used
12 provider = IBMQ.get_provider(hub='ibm-q')
13 print(provider.backends(operational=True))
14 backend = provider.get_backend('ibmq_vigo')
15
16 # Set prime p and angle to implement the mo1qfa to module p problem
17 mod = 11
18 theta = 4*np.pi/mod
19
20 # Set all instances to be tested
21 a = []
22 for x in range(0,91):
23     n = mod*x
```

```
24      a.append(n)
25      a.append(3+n)
26
27
28  # Initialize a quantum circuit list, a jobs list, a results list and list
    ↪  with the probabilities of each output state
29  qc = []
30  jobs = []
31  results = []
32  counts = []
33  # Initialize the number of executions of each experiment
34  nshots = 8192
35
36  # In case of restarting the process each experiment is being saved,
37  # here the program see which experiments have not been executed
38  nlines = 0
39  with open("results.txt","r+") as file:
40      while file.readline():
41          nlines += 1
42  a = a[nlines:]
43  print("faltam ",len(a))
44
45  # The main loop, that create, run, and save each experiment
46  for i,x in enumerate(a):
47
48      # Create a circuit with one qubit and one output bit
49      qc.append(QuantumCircuit(1, 1))
50
51      # Add X ports ry(theta) on qubit 0, where X is an instance of the
        ↪  experiment
52      for _ in range(x):
53          qc[i].ry(theta,0)
54
55      # Add the measure ports on qubit 0
56      qc[i].measure(0, 0)
57
58      # Run and monitors the job
59      jobs.append(execute(qc[i], backend, shots=nshots))
60      job_monitor(jobs[i])
61
62      # Gets the probability of each output state
63      results.append(jobs[i].result())
64      counts.append(results[i].get_counts(qc[i]))
65
66      # Saves the probabilities in case of stopping the experimento to restart
```

```
67      # from this instance latter
68      txt = str(x)
69      for key in counts[i]:
70          txt += " " + str(key) + " "+ str(counts[i][key]/nshots*100)
71      txt += "\n"
72      with open("results.txt","a+") as file:
73          file.write(txt)
74
75      # Prints progress
76      print(txt)
```

As linhas 1 à 5 importam pacotes necessários para o uso do QisKit e para aproximação mais fiel do ângulo.

As linhas 9 à 14 garantem a conexão e selecionam qual computador quântico do projeto IBMQ será usado.

As linhas 17 e 18 realizam configurações iniciais referentes ao problema, como o qual instancia de modulo que será calculado e o ângulo para tal (explicado na seção 4.2.2).

As linhas 21 à 25 definem as instancias a serem executadas: instancias com tamanho entre 0 e 1000 que são congruentes a 0 ou a 3 módulo 11.

As linhas 29 à 31 inicializam listas que manterão informações, respectivamente, sobre os circuitos quânticos, as tarefas a serem enviadas ao *ibmq_vigo*, os resultados obtidos e a probabilidade de saide de cada estado.

Na linha 32 é configurado que todas as instancias serão executadas 8192 vezes.

O sistema possui persistência, principalmente pela demora entre as execuções, dessa forma, nas linhas 38 à 43 é verificado quais instancias foram computadas e quais ainda faltam.

A linha 46 indica o loop principal, que cria, executa, e salva cada experimento.

A linha 49 instancia um circuito quântico com 1 qubit e 1 bit de output.

As linhas 52 à 56 constroem o circuito em si. As linhas 52 e 53 incluem as portas $Ry(\theta)$, a ser operada no qubit 0 (o único qubit desse circuito).

Onde o número de portas incluída é igual ao tamanho da instancia. A linha 56 adiciona a porta de medida, para colapsar o estado em $|0\rangle$ ou $|1\rangle$.

As linhas 59 enviam a tarefa aos servidores da IBM, que executaram o circuito especificado, o número de vezes especificada na plataforma desejada. Na linha 60 a tarefa enviada é monitorada, de forma a informar quantos outros circuitos estão na fila.

As linhas 63 e 64 pegam os resultados, e separam entre a probabilidade do resultado ser 0 ou 1.

As linhas 68 à 73 implementam a persistência, de forma a salvar qual a instancia e a probabilidade de cada estado de saída.

De forma a manter informação sobre o progresso, a linha 76 imprime o resultado obtido pela execução de cada circuito.

APÊNDICE B

Artigo

Modelos Computacionais Quânticos

Lucas Cavalcante de Sousa[1]
Orientador: Prof. Dr. Eduardo Inacio Duzzioni[1]
Coorientadora: Profª. Dra. Jerusa Marchi[1]

[1]Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brasil

lucas.sousa@grad.ufsc.br

Resumo. A computação quântica vem evoluindo bastante nos últimos anos. O presente trabalho estuda a computação quântica por meio da utilização de máquinas abstratas que utilizam efeitos quânticos: as versões quânticas de autômatos finitos e de pilha. Esse trabalho apresenta alguns dos modelos existentes e suas propriedades conhecidas. Apresenta-se também exemplos de linguagens tratáveis pelo MO-1QFA, um autômato finito quântico com menor poder de reconhecimento, que ainda assim, reconhece algumas linguagens que sua versão clásica não reconhece. Esse trabalho também apresenta um estudo de caso explorando os erros atrelados a execução de um autômato finito quântico em uma plataforma quântica real.

Palavras-chave:
Modelos computacionais, modelos de máquinas abstratas, computação quântica, modelos computacionais quânticos

1. Introdução

A computação quântica têm sua importância por 2 lados principais, a Ciência da Computação e a Física.

Do lado da computação a importância vem pela dificuldade de manter a lei de Moore como conjecturada em 1975 por Gordon. E. Moore que dizia que o número de transistores em um processador dobraria a cada 2 anos.

Hennessy e Patterson [Hennessy and Patterson 2018] analisaram o desenvolvimento de performance dos processadores de 1978 à 2018. Por meio desta análise percebemos que o crescimento descrito por Moore não ocorreu de fato, e mesmo com os vários avanços tecnológicos obtidos, ele não se manteve constante. Em [Hennessy and Patterson 2018] é possivel ver que de 2015 à 2018 o crescimento observado foi de 3,5% ao ano, dobrando a cada 20 anos. Incitando a necessidade de outras formas para manter a lei de Moore.

Do lado da Física, vem a necessidade exposta em 1982 por Feynman [Feynman 1982]: simular sistemas físicos quânticos em um computador universal - tarefa que se mostra difícil mesmo nas máquinas atuais, devido ao crescimento exponencial requerido para tal simulação.

Dessa forma, a computação quântica se mostra uma área importante de estudo. De maneira mais especifica, é importante também um estudo assim como

o ocorrido na computação clássica. Partindo do processamento de autômatos de forma a entender quais classes de problemas são tratáveis com tais dispositivos e como realizar tais operações. Assim o presente trabalho contribui:

- Exemplificando problemas tratáveis com um dos modelos de autômato finito quântico, o MO-1QFA;
- Analisando o erro do MO-1QFA em uma plataforma quântica real, através do IBMQ[IBM a].

2. qubit.tex

O qubit é o menor pedaço de informação quântica possível. Sua versão clássica o bit possui dois estados possíveis ou interpretações, como por exemplo: ligado ou desligado, "0" ou "1" e verdadeiro ou falso. Combinando mais bits podemos ter mais estados e, portanto, mais interpretações. Dessa forma, podemos guardar mais informação e processa-la de mais formas. Mais especificamente essa quantidade cresce de maneira exponencial com relação a quantidade de bits.

O bit quântico, qubit, [Nielsen and Chuang 2010] é um conceito análogo ao bit clássico. Ele representa o menor pedaço de informação que um sistema quântico pode representar. Os qubits, neste trabalho, serão objetos matemáticos, mas assim como os bits podem ser interpretados em sistemas físicos de formas variadas e implementados de diferentes formas.

Um qubit, assim como um bit, possui 2 estados como por exemplo $|0\rangle$ e $|1\rangle$, análogos ao "0" e "1" clássicos. Onde as bases $|0\rangle$ e $|1\rangle$ são vetores bidimensionais:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}; \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}; \tag{1}$$

Porém, existem mais estados possíveis para um qubit. Um qubit $|\psi\rangle$ arbitrário pode estar em uma combinação linear dos estados, uma superposição; desta forma, $|\psi\rangle$ é descrito por:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{2}$$

Podemos operar a vontade com $|\psi\rangle$ porem, quando quisermos realizar saber o resultado "0" ou "1" precisamos realizar uma medida. A mecânica quântica nos garante que no momento da medida teremos como resultado da leitura de $|\psi\rangle$, o estado $|0\rangle$ com probabilidade $|\alpha|^2$ ou o estado $|1\rangle$ com probabilidade $|\beta|^2$, onde $|\alpha|^2$ e $|\beta|^2$ somam 1, como manda a probabilidade.

Representações de um Qubit

Também é possível representar um qubit de forma gráfica, através da esfera de Bloch (fig. 1). Na superfície da esfera de Bloch podemos ver todos os possíveis estados de um qubit.

Um qubit pode estar em qualquer ponto da esfera de Bloch, portanto possui infinitas possibilidades de estados, o que pode facilmente nos fazer pensar que um qubit pode armazenar informação infinita. Porém, como ao medir um qubit só temos duas possíveis respostas, não temos realmente informação infinita armazenadas em

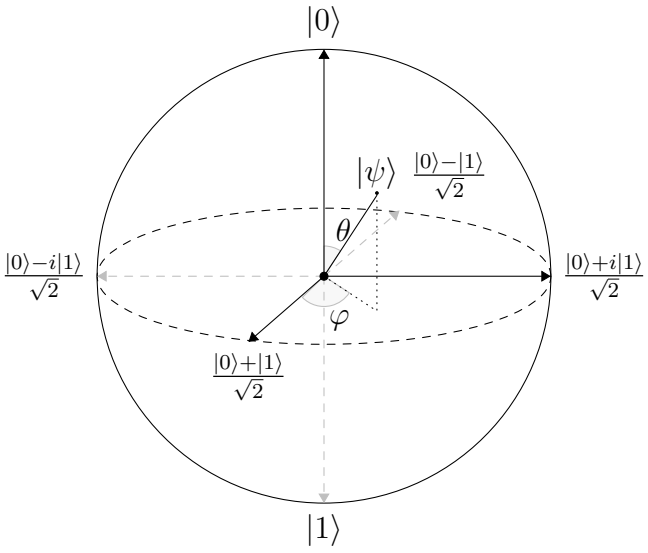

Figura 1. Representação de um qubit na esfera de Bloch.

um qubit. Por outro lado, antes da medida, realmente podemos ter mais informação do que $|0\rangle$ e $|1\rangle$ como veremos posteriormente.

Além disso, é possível alterar qualquer estado da esfera, para qualquer outro, também na esfera, ao multiplica-lo pela seguinte matrix, com os ângulos bem configurados:

$$U = \begin{bmatrix} cos\left(\frac{\theta}{2}\right) & -e^{i\gamma}sin\left(\frac{\theta}{2}\right) \\ e^{i\phi}sin\left(\frac{\theta}{2}\right) & e^{i\gamma+i\phi}cos\left(\frac{\theta}{2}\right) \end{bmatrix}. \tag{3}$$

O MO1QFA é o modelo mais simples de autômato finito quântico e que possui a menor expressividade. Esse modelo foi proposto por Moore C. e Crutchfield J. [Moore and Crutchfield 2000] em 2000, com o intuito de adicionar os efeitos quânticos ao autômato finito, tais como: transições dadas por transformações unitárias e a necessidade de se medir o sistema para obter alguma informação. A sigla MO1QFA vêm de measure-once 1-way quantum finite automata, nome obtido dado algumas características do seu funcionamento:

- o estado quântico é medido uma única vez e sempre ao final da execução, por isto é chamado de measure-once;
- em comparação a outras versões de autômatos que podem mover o cabeçote de leitura para ambos os lados, o MO1QFA é chamado de 1-way, já que seu cabeçote de leitura se move em uma única direção, sempre em direção ao fim da palavra. No início da computação o cabeçote está no primeiro caractere da palavra e, a cada transição, avança um caractere até o fim da palavra.

Definição 1. O MO1QFA é dado pela quíntupla $M = (Q, \Sigma, \delta, |q_0\rangle, F)$ onde:

$Q = \{|q_0\rangle, \ldots, |q_n\rangle\}$ é um conjunto finito de estados - Q é contido por um espaço de Hilbert;

$\Sigma = \{\sigma_0, \ldots, \sigma_k\}$ é um conjunto finito do alfabeto de entrada;

δ é um conjunto de matrizes unitárias, U_σ, que descreve as transições entre estados para cada símbolo σ do alfabeto de entrada;

$|q_0\rangle \in Q$ e é o estado inicial do autômato;
$F \subseteq Q$ é o conjunto de estados finais.

A computação de uma palavra w pelo MO1QFA é dada por:

$$f(w) = \|P_{acc} U_w |q_0\rangle\|^2, \tag{4}$$

onde U_w é operar do caractere inicial ao final da palavra:

$$U = U_{w_{|w|-1}} \dots U_{w_1} U_{w_0}, \tag{5}$$

e P_{acc} é o operador de projeção conjunto de estados finais do autômato:

$$P_{acc} = \sum_{q_i \in F} |q_i\rangle\langle q_i| . \tag{6}$$

f(w) representa a probabilidade de M aceitar w.

3. Exemplos de Problemas Tratáveis com o MO1QFA

Nesta seção serão apresentados 2 problemas e como eles são tratados com o MO1QFA. Ambos exemplos possuem uma estrutura muito semelhante. Possuem 2 estados $|0\rangle e |1\rangle$ e utilizam o conjunto de superposições a seu favor, mais especificamente os modelos realizam somente rotações atraves do eixo y, usando a seguinte matriz de transição:

$$U_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}. \tag{7}$$

Dessa forma, o novo conjunto de superposições de estados resulta numa esfera (fig. 2).

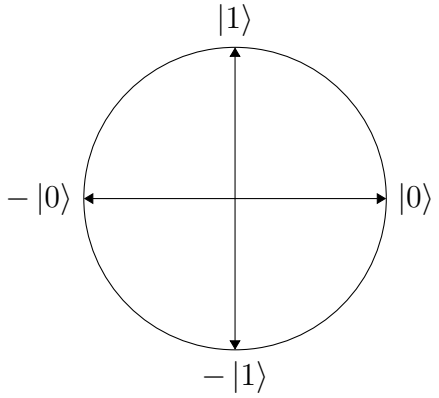

Figura 2. Representação de um qubit considerando somente duas dimensões.

Ao aplicar a matriz da eq. (7) com um θ positivo resulta numa rotação no sentido anti horário e com um θ negativo resulta em uma rotação no sentido horário.

3.1. $L = \{(a, b)^* \mid \#a's \neq \#b's\}$

Say e Yakaryılmaz [Say and Yakaryılmaz 2014] propuseram que, com MO1QFA não determinístico podemos resolver alguns problemas da classe das linguagens livres de contexto (o conjunto de linguagens reconhecíveis por autômatos de pilha não determinísticos[Sipser 2012]), o que é impossível com autômatos finitos clássicos[Sipser 2012], para isso, eles estudaram a seguinte linguagem:

$$L = \{(a + b)^* \mid \#a's \neq \#b's\}. \tag{8}$$

Um MO1QFA pode facilmente reconhecer tal linguagem, de forma que o autômato possui 0% de chance de aceitar palavras não pertencentes a linguagem e aceita as palavras da linguagem com uma probabilidade maior que 0%.

A ideia neste autômato é utilizarmos do fato que o conjunto de possibilidades formam um círculo, e o fato de que um círculo possui infinitos pontos. Assim ao utilizar as seguintes matrizes:

$$A = \begin{bmatrix} \cos(-\sqrt{2}\pi) & \sin(-\sqrt{2}\pi) \\ \sin(-\sqrt{2}\pi) & \cos(-\sqrt{2}\pi) \end{bmatrix}, B = \begin{bmatrix} \cos(\sqrt{2}\pi) & \sin(\sqrt{2}\pi) \\ \sin(\sqrt{2}\pi) & \cos(\sqrt{2}\pi) \end{bmatrix}, \tag{9}$$

com θ sendo um múltiplo irracional de π as rotações descritas na eq. (9) descrevem uma circunferência densa, é possível receber infinitos a's ou b's sem nunca passar por um ponto já visitado da circunferência. Dessa forma, podemos usar a superposição dos estados para realizar uma contagem, saindo do estado $|0\rangle$ sempre que recebemos um a realizamos uma rotação no sentido anti-horário, e quando recebemos um b realizamos uma rotação no sentido horário, como os ângulos das matrizes A e B são inversos, uma desfaz as rotações da outra. Assim, caso o número de a's e b's sejam iguais ainda estaremos no estado inicial e não aceitamos a palavra. De forma mais visual, o diagrama da fig. 3 descreve o autômato.

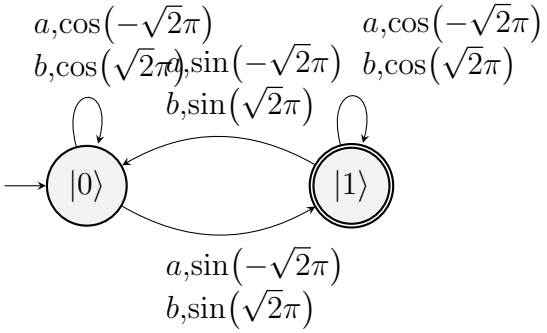

Figura 3. Autômato que reconhece uma palavra com número de a's e b's diferentes.

Esse MO1QFA é dito não determinístico pelo fato de reconhece com ponto de corte 0 todas as palavras da linguagem, ou seja, com probabilidade maior que 0 reconhece as palavras da linguagem, e com probabilidade igual a 0 reconhece as palavras que não pertencem a linguagem. O que pode ser um dificuldade para seu uso já que possui falsos positivos. Importante restar que tal autômato usa a superposição dos estados como uma memória infinita e reconhece uma linguagem que classicamente necessitaria de um autômato de pilha.

3.2. $\boldsymbol{L = \{a^* \mid 0 \equiv \#a's (\text{mod } p)\}}$

Say e Yakaryılmaz [Say and Yakaryılmaz 2014] propuseram que, com MO1QFA podemos resolver problemas que possuem erro limitado utilizando menos estados quando comparado à mesma solução em um autômato finito clássico, e para demonstrar isso estudaram o problema do módulo de um número primo:

$$\text{MOD}^p = \{a^{jp} | \text{ j é um inteiro não negativo}\} \tag{10}$$

mo1qfa O autômato apresentado possui somente 2 estados, e reconhece valores congruentes a 0 modulo de p com 100% de chande, porém, possibilita falsos positivos, o limite superior do erro se encontra nos valores que tendem a congruente a $p/2$ modulo p, onde o erro tende a:

$$\cos^2\left(\frac{\pi}{p}\right). \tag{11}$$

Para realizar esse autômato podemos dividir o circulo de possibilidades da fig. 2 em p partes, onde cada corte representa os possíveis valores de congruência, $0, 1, 2, \ldots, p-1$. Para tal utilizamos a seguinte matriz de rotação:

$$A = \begin{bmatrix} \cos\left(\frac{2\pi}{p}\right) & \sin\left(\frac{2\pi}{p}\right) \\ \sin\left(\frac{2\pi}{p}\right) & \cos\left(\frac{2\pi}{p}\right) \end{bmatrix}, \tag{12}$$

Que "corta" a circunferência em p partes, e rotaciona o estado de acordo. De forma mais visual, o diagrama da fig. 4 descreve o autômato. Importante ressaltar que

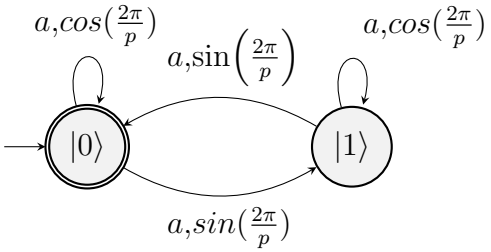

Figura 4. Autômato que reconhece uma palavra com número de a's e b's diferentes.

o autômato apresentado pode ser facilmente convertido para reconhecer modulo de qualquer primo, para fazer isso de maneira clássica, seriam necessários p estados, enquanto neste MO1QFA são necessários somente 2. Em contra partida, o para os valores que tendem a congruente a $p/2$ modulo p (eq. (11)) cresce muito rápido, possibilitando que estes valores sejam equivocadamente aceitos em mais 80% já para primos maiores de 7.

4. Experimentos

Foram realizados alguns experimentos com o MO1QFA que resolve o problema do modulo 11 na nas plataformas quânticas da IBM através do IBMQ[IBM b] de forma a avaliar a execução de autômatos quânticos em plataformas quânticas atuais.

O IBMQ utilizam o modelo circuital[Experience a] de computação quântica, isto é, toda a computação é descrita por uma sequência de portas quânticas, onde cada porta representa uma matriz de rotação no qubit especificado. Portanto, é necessário primeiro mapear o MO1QFA para seguir as descrições do modelo circuital.

Como o modelo apresentado seção 3.2 possui somente 2 estados, o circuito utilizado terá somente 1 qubit. As transições serão realizadas através da porta $R_y(\theta)$ que realiza uma rotação de θ em relação ao eixo y. Porém dado a forma como a porta R_y é descrita e como a matriz de rotação da eq. (7) é descrita, é necessário utilizar portas R_y com ângulo 2θ onde θ é corresponde a $\frac{2\pi}{11}$ de acordo com a descrição da matriz de transição do autômato (eq. (12)).

Ou seja, o mapeamento do MO1QFA para o modelo circuital do QisKit se dá por uma sequência de portas $R_y(\frac{4\pi}{11})$ para descrever as rotações aplicadas pelas transições tomadas durante execução de uma entrada no autômato.

O experimento realizado utilizou a plataforma ibmq_vigo que possui 5 qubits, mas dado a natureza do autômato só foi utilizado o qubit 0, que possui taxa de erro aproximado de 1.16×10^{-3} em suas rotações de um qubit. Foram testados os valores congruentes a 0 e a 3 módulo 11 no intervalo de 0 à 1000, portanto 182 valores destintos, onde cada um foi executado 8192 vezes.

Teoricamente é esperado que os valores congruentes a 0 módulo 11 tivessem 100% de probabilidade de serem aceitos e os valores congruentes a 3 módulo 11 tivessem 2,0253% de probabilidade de serem aceitos.

Na fig. 5 podemos ver todos os casos analisados, o valor de erro é alto (com uma média de aproximadamente 7.579 pontos) e ocorre de maneira uniforme durante todo o intervalo (com um desvio padrão de aproximadamente 1.98), ou seja, parece ignorar a decoerência que deveria ocorrer na aplicação de uma quantidade alta de portas, como é o caso de aplicar centenas de portas. Para melhor analise as ?? e fig. 7 separam os valores congruentes a 0 e 3 modulo 11.

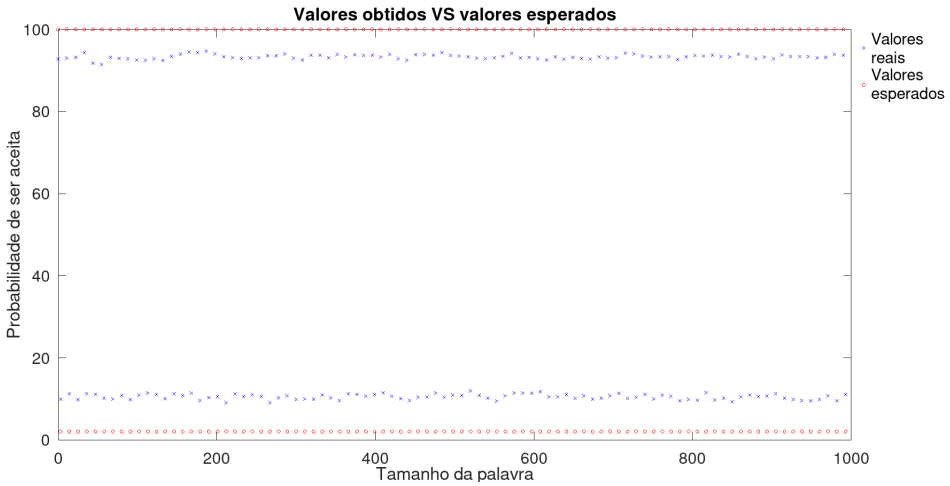

Figura 5. Comparação das probabilidades esperadas, em vermelho, e a probabilidade obtida, em azul. Os valores de cima são referentes aos tamanhos congruentes a 0 módulo 11, e os de baixo aos tamanhos congruentes a 3 módulo 11.

Erro máximo	9.900918211974883	Tamanho da entrada	520
Erro mínimo	5.2978515625	Tamanho da entrada	187
Média		7.578928049153639	
Variância		1.205517567315090	
Desvio padrão		1.097960640148403	

Tabela 1. Resumo dos valores de erro absoluto obtidos.

Na fig. 6 e tabela 2 podemos ver os casos congruentes a 0. Aqui o erro ainda é alto, porém ligeiramente menor (com uma média de aproximadamente 6.665 pontos, quase um ponto a menos que a média geral) e ocorre de maneira uniforme (com um desvio padrão de aproximadamente 0.559).

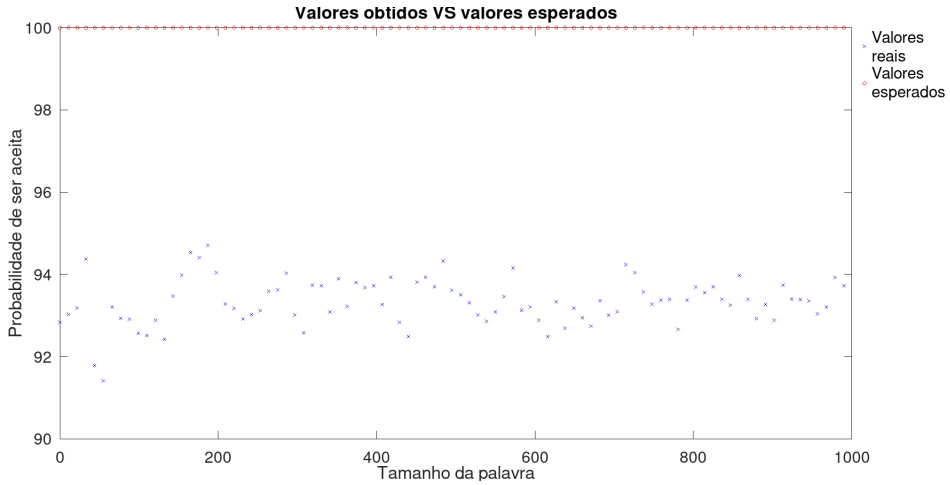

Figura 6. Comparação das probabilidades esperadas, em vermelho, e a probabilidade obtida, em azul. Os valores são referentes ás entradas com tamanho congruente a 0 módulo 11.

Erro máximo	8.58154296875	Tamanho da entrada	55
Erro mínimo	5.2978515625	Tamanho da entrada	187
Média		6.664502489697802	
Variância		0.3123444256150856	
Desvio padrão		0.5588778270920091	

Tabela 2. Resumo dos valores de erro absoluto obtidos para entradas com tamanho congruente a 0 módulo 11.

Na fig. 7 e tabela 3 podemos ver os casos congruentes a 3. Aqui o erro é mais alto que no caso dos congruentes a 0 (com uma média de aproximadamente 8.49 pontos, quase um ponto a mais que a média geral e quase dois pontos a mais que no caso dos congruentes a 0) e ocorre de maneira uniforme (com um desvio padrão de aproximadamente 0.653, ligeiramente maior que no caso dos congruentes a 0).

Para entender os fatores que compõem o erro que ocorrem durante a computação foi definido uma função para aproximar os valores obtidos do valor esperado, dessa forma talvez se possa entender quais erros estão ocorrendo, e qual o impacto de cada um.

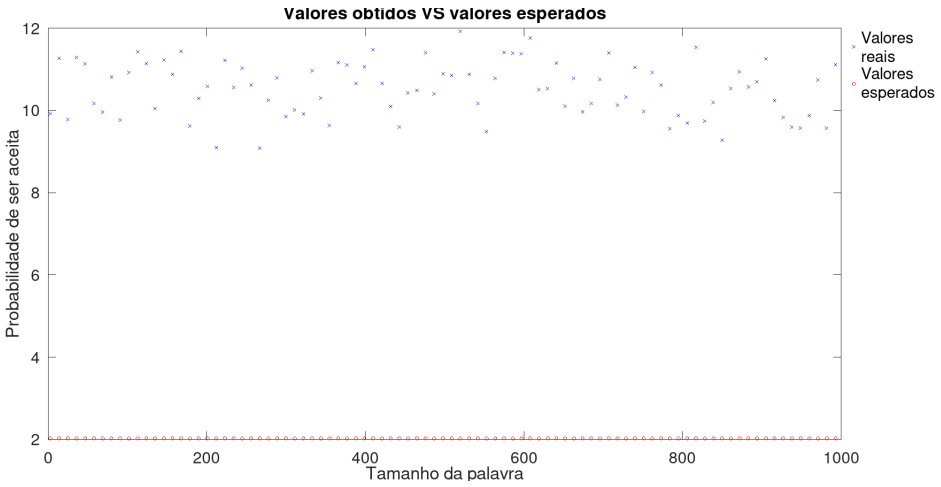

Figura 7. Comparação das probabilidades esperadas, em vermelho, e a probabilidade obtida, em azul. Os valores são referentes às entradas com tamanho congruente a 3 módulo 11.

Erro máximo	9.900918211974883	Tamanho da entrada	520
Erro mínimo	7.056679930724883	Tamanho da entrada	267
Média		8.493353608609491	
Variância		0.4263425014428133	
Desvio padrão		0.6529490802833046	

Tabela 3. Resumo dos valores de erro absoluto obtidos para entradas com tamanho congruente a 3 módulo 11.

O erro pode ser definido por 3 fatores principais: α (representa um erro de amplitude), δ (representa um erro de fase) e β (um ruído):

$$\alpha \cos^2\left(\frac{2\pi}{11}|w| + \delta\right) + \beta, \tag{13}$$

quando $\alpha = 1$, $\delta = 0$ e $\beta = 0$ não ocorre nenhum erro.

Nesse estudos foram abordados os erros relacionados a amplitude e ao ruido, portanto é importante entender qual a relação deles com a função sem erro: valores maiores que 1 de α espicham a função, enquanto menores encolhem ela; valores positivos de β elevam a função com relação ao eixo y, enquanto valores negativos abaixam a função original.

Os valores para α e β podem ser obtidos analiticamente a partir dos erros já apresentados nas tabelas 1 a 3. β pode ser definido como o erro médio dos valores com tamanho congruentes a 3 módulo 11, o que "levanta" a função ao ser somado. E α pode ser definido pela seguinte equação:

$$\alpha = (100 - E_0 - \beta)/100 \approx 0.848, \tag{14}$$

onde E_0 representa o erro médio dos valores com tamanho congruente 0 módulo 11.

Podemos ver na tabela 4 que os valores selecionados para α e β realmente aproximam a função, demonstrando se aderir bem a função original.

Média	0.51903759262535
Variância	0.1470678913781397
Desvio padrão	0.3834943172696822

Tabela 4. Resumo dos valores de erro obtidos após a função de aproximação com valores α e β ótimos.

Além disso, essas grandezas de erro demonstram que aproximadamente 25% da amplitude é perdida mesmo para execuções iniciais com um número pequeno de a's, e se mantem constante mesmo para quantidades de a's próximos a 1000, demonstrando algo que parece ignorar a decoerência que deveria ocorrer na aplicação de uma quantidade alta de portas, como é o caso de aplicar centenas de portas.

5. Considerações Finais

O presente trabalho apresentou o MO-1QFA e demonstrou problemas tratáveis ele, o modelo mais simples de QFA, além de demonstrar algumas diferenças de implementação entre AF's e o QFA's. Entre os exemplos, demonstrou que os QFA's resolvem algumas classes de problemas utilizando menos memória, embora para isso haja um sacrifício na certeza da resposta. E que, além disso, os QFA's extrapolam a classe das Linguagens Regulares, mostrando que eles podem realizar tarefas que AF's não podem. O presente trabalho também trouxe um estudo sobre a implementação de um problema resolvido por meio de um autômato finito quântico em um computador quântico real, e discutiu sobre as capacidades atuais de implementação deste modelo.

6. Trabalhos Futuros

É importante estudar outros modelos de autômatos quânticos, além do MO-1QFA, para se ter um entendimento sobre como podem ser utilizados, de forma a obtermos um maior detalhamento sobre como desenvolvê-os para resolver uma dada linguagem, em especial os autômatos de pilha e como funciona o seu uso de memória. Outro assunto a se abordar é com relação aos testes na plataforma da IBM, é importante um estudo mais aprofundado sobre os causadores dos erros, e de testes em outras plataformas para comparação dos resultados. Seria interessante também a implementação de mais autômatos e o estudo de seu mapeamento para o modelo circuital de computação. Espera-se que com o estudo desse erro, possa se estudar sua mitigação. Espera-se que esse trabalho sirva de inspiração para mais estudos com relação a versões quânticas de autômatos finitos e de pilha, além do estudo das suas classes de complexidade.

Referências

Abraham, H., Akhalwaya, I. Y., Aleksandrowicz, G., Alexander, T., Alexandrowics, G., Arbel, E., Asfaw, A., Azaustre, C., Barkoutsos, P., Barron, G., Bello, L., Ben-Haim, Y., Bevenius, D., Bishop, L. S., Bosch, S., Bucher, D., CZ, Cabrera, F., Calpin, P., Capelluto, L., Carballo, J., Carrascal, G., Chen, A., Chen, C.-F., Chen, R., Chow, J. M., Claus, C., Clauss, C., Cross, A. J., Cross, A. W., Cruz-Benito, J., Cryoris, Culver, C., Córcoles-Gonzales, A. D., Dague, S., Dartiailh,

M., Davila, A. R., Ding, D., Dumitrescu, E., Dumon, K., Duran, I., Eendebak, P., Egger, D., Everitt, M., Fernández, P. M., Frisch, A., Fuhrer, A., GOULD, I., Gacon, J., Gadi, Gago, B. G., Gambetta, J. M., Garcia, L., Garion, S., Gawel-Kus, Gomez-Mosquera, J., de la Puente González, S., Greenberg, D., Gunnels, J. A., Haide, I., Hamamura, I., Havlicek, V., Hellmers, J., Herok, Ł., Horii, H., Howington, C., Hu, S., Hu, W., Imai, H., Imamichi, T., Iten, R., Itoko, T., Javadi-Abhari, A., Jessica, Johns, K., Kanazawa, N., Karazeev, A., Kassebaum, P., Kovyrshin, A., Krishnan, V., Krsulich, K., Kus, G., LaRose, R., Lambert, R., Latone, J., Lawrence, S., Liu, D., Liu, P., Mac, P. B. Z., Maeng, Y., Malyshev, A., Marecek, J., Marques, M., Mathews, D., Matsuo, A., McClure, D. T., McGarry, C., McKay, D., Meesala, S., Mezzacapo, A., Midha, R., Minev, Z., Mooring, M. D., Morales, R., Moran, N., Murali, P., Müggenburg, J., Nadlinger, D., Nannicini, G., Nation, P., Naveh, Y., Nick-Singstock, Niroula, P., Norlen, H., O'Riordan, L. J., Ollitrault, P., Oud, S., Padilha, D., Paik, H., Perriello, S., Phan, A., Pistoia, M., Pozas-iKerstjens, A., Prutyanov, V., Pérez, J., Quintiii, Raymond, R., Redondo, R. M.-C., Reuter, M., Rodríguez, D. M., Ryu, M., Sandberg, M., Sathaye, N., Schmitt, B., Schnabel, C., Scholten, T. L., Schoute, E., Sertage, I. F., Shammah, N., Shi, Y., Silva, A., Siraichi, Y., Sivarajah, S., Smolin, J. A., Soeken, M., Steenken, D., Stypulkoski, M., Takahashi, H., Taylor, C., Taylour, P., Thomas, S., Tillet, M., Tod, M., de la Torre, E., Trabing, K., Treinish, M., TrishaPe, Turner, W., Vaknin, Y., Valcarce, C. R., Varchon, F., Vogt-Lee, D., Vuillot, C., Weaver, J., Wieczorek, R., Wildstrom, J. A., Wille, R., Winston, E., Woehr, J. J., Woerner, S., Woo, R., Wood, C. J., Wood, R., Wood, S., Wootton, J., Yeralin, D., Yu, J., Zdanski, L., Zoufalc, anedumla, azulehner, bcamorrison, brandhsn, dennis-liu 1, drholmie, elfrocampeador, fanizzamarco, gruu, kanejess, klinvill, lerongil, ma5x, merav aharoni, mrossinek, ordmoj, strickroman, tigerjack, yang.luh, and yotamvakninibm (2019). Qiskit: An open-source framework for quantum computing.

Abraham, H., Akhalwaya, I. Y., Aleksandrowicz, G., Alexander, T., Alexandrowics, G., Arbel, E., Asfaw, A., Azaustre, C., Barkoutsos, P., Barron, G., Bello, L., Ben-Haim, Y., Bevenius, D., Bishop, L. S., Bosch, S., Bucher, D., CZ, Cabrera, F., Calpin, P., Capelluto, L., Carballo, J., Carrascal, G., Chen, A., Chen, C.-F., Chen, R., Chow, J. M., Claus, C., Clauss, C., Cross, A. J., Cross, A. W., Cruz-Benito, J., Cryoris, Culver, C., Córcoles-Gonzales, A. D., Dague, S., Dartiailh, M., Davila, A. R., Ding, D., Dumitrescu, E., Dumon, K., Duran, I., Eendebak, P., Egger, D., Everitt, M., Fernández, P. M., Frisch, A., Fuhrer, A., GOULD, I., Gacon, J., Gadi, Gago, B. G., Gambetta, J. M., Garcia, L., Garion, S., Gawel-Kus, Gomez-Mosquera, J., de la Puente González, S., Greenberg, D., Gunnels, J. A., Haide, I., Hamamura, I., Havlicek, V., Hellmers, J., Herok, Ł., Horii, H., Howington, C., Hu, S., Hu, W., Imai, H., Imamichi, T., Iten, R., Itoko, T., Javadi-Abhari, A., Jessica, Johns, K., Kanazawa, N., Karazeev, A., Kassebaum, P., Kovyrshin, A., Krishnan, V., Krsulich, K., Kus, G., LaRose, R., Lambert, R., Latone, J., Lawrence, S., Liu, D., Liu, P., Mac, P. B. Z., Maeng, Y., Malyshev, A., Marecek, J., Marques, M., Mathews, D., Matsuo, A., McClure, D. T., McGarry, C., McKay, D., Meesala, S., Mezzacapo, A., Midha, R., Minev, Z., Mooring, M. D., Morales, R., Moran, N., Murali, P., Müggenburg, J., Nadlinger, D., Nannicini, G., Nation, P., Naveh, Y., Nick-Singstock, Niroula, P., Norlen,

H., O'Riordan, L. J., Ollitrault, P., Oud, S., Padilha, D., Paik, H., Perriello, S., Phan, A., Pistoia, M., Pozas-iKerstjens, A., Prutyanov, V., Pérez, J., Quintiii, Raymond, R., Redondo, R. M.-C., Reuter, M., Rodríguez, D. M., Ryu, M., Sandberg, M., Sathaye, N., Schmitt, B., Schnabel, C., Scholten, T. L., Schoute, E., Sertage, I. F., Shammah, N., Shi, Y., Silva, A., Siraichi, Y., Sivarajah, S., Smolin, J. A., Soeken, M., Steenken, D., Stypulkoski, M., Takahashi, H., Taylor, C., Taylour, P., Thomas, S., Tillet, M., Tod, M., de la Torre, E., Trabing, K., Treinish, M., TrishaPe, Turner, W., Vaknin, Y., Valcarce, C. R., Varchon, F., Vogt-Lee, D., Vuillot, C., Weaver, J., Wieczorek, R., Wildstrom, J. A., Wille, R., Winston, E., Woehr, J. J., Woerner, S., Woo, R., Wood, C. J., Wood, R., Wood, S., Wootton, J., Yeralin, D., Yu, J., Zdanski, L., Zoufalc, anedumla, azulehner, bcamorrison, brandhsn, dennis-liu 1, drholmie, elfrocampeador, fanizzamarco, gruu, kanejess, klinvill, lerongil, ma5x, merav aharoni, mrossinek, ordmoj, strickroman, tigerjack, yang.luh, and yotamvakninibm. Qiskit - class ry gate. Disponivel em: https://qiskit.org/documentation/api/qiskit.extensions.RYGate.html?highlight=ry%20gate#qiskit.extensions.RYGate. Acessado em 2019-10-22.

Amano, M. and Iwama, K. (1999). Undecidability on quantum finite automata. In Proceedings of the thirty-first annual ACM symposium on Theory of computing, pages 368–375. ACM.

Ambainis, A., Beaudry, M., Golovkins, M., Ķikusts, A., Mercer, M., and Thérien, D. (2004). Algebraic results on quantum automata. In Annual Symposium on Theoretical Aspects of Computer Science, pages 93–104. Springer.

Ambainis, A. and Freivalds, R. (1998). 1-way quantum finite automata: strengths, weaknesses and generalizations. In Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No. 98CB36280), pages 332–341. IEEE.

Ambainis, A. and Ķikusts, A. (2003). Exact results for accepting probabilities of quantum automata. Theoretical Computer Science, 295(1-3):3–25.

Ambainis, A., KĶikusts, A., and Valdats, M. (2001). On the class of languages recognizable by 1-way quantum finite automata. In Annual Symposium on Theoretical Aspects of Computer Science, pages 75–86. Springer.

Ambainis, A., Nayak, A., Ta-Shma, A., and Vazirani, U. (1998). Dense quantum coding and a lower bound for 1-way quantum automata. arXiv preprint quant-ph/9804043.

Ambainis, A. and Watrous, J. (2002). Two-way finite automata with quantum and classical states. Theoretical Computer Science, 287(1):299–311.

Ambainis, A. and Yakaryılmaz, A. (2012). Superiority of exact quantum automata for promise problems. Information Processing Letters, 112(7):289–291.

Ambainis, A. and Yakaryılmaz, A. (2015). Automata and quantum computing. arXiv preprint arXiv:1507.01988.

Amdahl, G. M. (1967). Validity of the single processor approach to achieving large scale computing capabilities. In Proceedings of the April 18-20, 1967, spring joint computer conference, pages 483–485. ACM.

Bertoni, A. and Carpentieri, M. (2001). Analogies and differences between quantum and stochastic automata. Theoretical Computer Science, 262(1-2):69–81.

Bertoni, A., Mereghetti, C., and Palano, B. (2003). Quantum computing: 1-way quantum automata. In International Conference on Developments in Language Theory, pages 1–20. Springer.

Bhatia, A. S. and Kumar, A. (2019). Quantum finite automata: survey, status and research directions. arXiv preprint arXiv:1901.07992.

Brodsky, A. and Pippenger, N. (2002). Characterizations of 1-way quantum finite automata. SIAM Journal on Computing, 31(5):1456–1478.

Dennard, R. H., Gaensslen, F. H., Rideout, V. L., Bassous, E., and LeBlanc, A. R. (1974). Design of ion-implanted mosfet's with very small physical dimensions. IEEE Journal of Solid-State Circuits, 9(5):256–268.

Deutsch, D. (1985). Quantum theory, the church–turing principle and the universal quantum computer. Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences, 400(1818):97–117.

Experience, I. Q. Introduction to quantum circuits - getting started with quantum circuits. Disponivel em https://quantum-computing.ibm.com/support/guides/introduction-to-quantum-circuits?section=5cae613866c1694be21df8cc. Acessado em: 2019-10-22.

Experience, I. Q. Introduction to quantum circuits - quantum gates. Disponivel em: https://quantum-computing.ibm.com/support/guides/introduction-to-quantum-circuits?page=5cae6f7735dafb4c01214bbe#other-single-qubit-gates. Acesso em: 22/10/2019.

Feynman, R. P. (1982). Simulating physics with computers. International journal of theoretical physics, 21(6):467–488.

Garzon, M. H. and Deaton, R. J. (1999). Biomolecular computing and programming. IEEE Transactions on Evolutionary Computation, 3(3):236–250.

Golovkins, M. (2000). Quantum pushdown automata. In International Conference on Current Trends in Theory and Practice of Computer Science, pages 336–346. Springer.

Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. arXiv preprint quant-ph/9605043.

Gudder, S. (2000). Quantum computers. International Journal of Theoretical Physics, 39(9):2151–2177.

Hennessy, J. L. and Patterson, D. A. (2018). Computer architecture: a quantitative approach. Elsevier.

Hirvensalo, M. (2010). Quantum automata with open time evolution. International Journal of Natural Computing Research (IJNCR), 1(1):70–85.

Hopcroft, J. E. (2008). Introduction to automata theory, languages, and computation. Pearson Education India.

IBM. Ibm q experience. Disponivel em : https://www.ibm.com/quantum-computing/technology/experience/. Acessado em: 2019-10-22.

IBM. Ibm quantum computing. Disponivel em: https://www.ibm.com/quantum-computing/. Acessado em: 2019-10-22.

Kikusts, A. (1998). A small 1-way quantum finite automaton. arXiv preprint quant-ph/9810065.

Kondacs, A. and Watrous, J. (1997). On the power of quantum finite state automata. In Proceedings 38th Annual Symposium on Foundations of Computer Science, pages 66–75. IEEE.

Koshiba, T. (2001). Polynomial-time algorithms for the equivalence for one-way quantum finite automata. In International Symposium on Algorithms and Computation, pages 268–278. Springer.

Li, L. and Qiu, D. (2006). A polynomial-time algorithm for the equivalence between quantum sequential machines. arXiv preprint quant-ph/0604085.

Li, L. and Qiu, D. (2008). Determining the equivalence for one-way quantum finite automata. Theoretical Computer Science, 403(1):42–51.

Li, L., Qiu, D., Zou, X., Li, L., Wu, L., and Mateus, P. (2012). Characterizations of one-way general quantum finite automata. Theoretical Computer Science, 419:73–91.

Lin, Y.-M., Valdes-Garcia, A., Han, S.-J., Farmer, D. B., Meric, I., Sun, Y., Wu, Y., Dimitrakopoulos, C., Grill, A., Avouris, P., et al. (2011). Wafer-scale graphene integrated circuit. Science, 332(6035):1294–1297.

Miller, V. S. (1985). Use of elliptic curves in cryptography. In Conference on the theory and application of cryptographic techniques, pages 417–426. Springer.

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

Moore, G. E. et al. (1965). Cramming more components onto integrated circuits.

Moore, G. E. et al. (1975). Progress in digital integrated electronics. In Electron Devices Meeting, volume 21, pages 11–13.

Mustafa, N. Complexity theory - randimized computation. Disponivel em: https://perso.esiee.fr/~mustafan/TechnicalWritings/Complexityslides/lec14.pdf. Acessado em: 2019-06-03.

Nakanishi, M. (2004). On the power of one-sided error quantum pushdown automata with classical stack operations. In International Computing and Combinatorics Conference, pages 179–187. Springer.

Nielsen, M. A. and Chuang, I. L. (2010). Quantum Computation and Quantum Information. Cambridge University Press, Cambridge, 10th anniversary edition edition.

Paschen, K. (2000). Quantum finite automata using ancilla qubits. university of karlsruhe. Technical report, Technical report.

Qiu, D. (2004). Automata theory based on quantum logic: some characterizations. Information and Computation, 190(2):179–195.

Qiu, D. (2007). Automata theory based on quantum logic: Reversibilities and pushdown automata. Theoretical Computer Science, 386(1-2):38–56.

Qiu, D. (2008). Some observations on two-way finite automata with quantum and classical states. In International Conference on Intelligent Computing, pages 1–8. Springer.

Qiu, D. and Li, L. (2008). An overview of quantum computation models: quantum automata. Frontiers of Computer Science in China, 2(2):193–207.

Qiu, D., Mateus, P., and Sernadas, A. (2009). One-way quantum finite automata together with classical states. arXiv preprint arXiv:0909.1428, pages 3006–3017.

Qiu, D. and Ying, M. (2004). Characterizations of quantum automata. Theoretical computer science, 312(2-3):479–489.

Rabin, M. O. (1963). Probabilistic automata. Information and control, 6(3):230–245.

Rivest, R. L., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2):120–126.

Roetteler, M., Naehrig, M., Svore, K. M., and Lauter, K. (2017). Quantum resource estimates for computing elliptic curve discrete logarithms. In International Conference on the Theory and Application of Cryptology and Information Security, pages 241–270. Springer.

Salomaa, A. (1976). Seymour ginsburg. algebraic and automata-theoretic properties of formal languages. fundamental studies in computer science, vol. 2. northholland publishing company, amsterdam and oxford, and american elsevier publishing company, inc., new york, 1975, xii+ 313 pp. The Journal of Symbolic Logic, 41(4):788–789.

Say, A. C. and Yakaryılmaz, A. (2014). Quantum finite automata: A modern introduction. In Computing with New Resources, pages 208–222. Springer.

Shor, P. W. (1994). Algorithms for quantum computation: Discrete logarithms and factoring. In Proceedings 35th annual symposium on foundations of computer science, pages 124–134. Ieee.

Sipser, M. (2012). Introduction to the Theory of Computation. Cengage Learning.

Wang, J. (2012). Handbook of Finite State Based Models and Applications. CRC press.

Yakaryilmaz, A. and Say, A. (2009a). Languages recognized by nondeterministic quantum finite automata. arXiv preprint arXiv:0902.2081.

Yakaryilmaz, A. and Say, A. C. (2009b). Languages recognized with unbounded error by quantum finite automata. In International Computer Science Symposium in Russia, pages 356–367. Springer.

Yakaryılmaz, A. and Say, A. C. (2011). Unbounded-error quantum computation with small space bounds. Information and Computation, 209(6):873–892.

Ying, M. (2000). Automata theory based on quantum logic ii. International Journal of Theoretical Physics, 39(11):2545–2557.