

**Título:** Um Algoritmo para Transformar Autômatos Finitos Não-Determinísticos em Autômatos Finitos Quânticos Preservando o Número de Estados e a Linguagem Reconhecida

**Autor:** Cheyenne Ribeiro Guedes Isidro ([cha@dsc.ufcg.edu.br](mailto:cha@dsc.ufcg.edu.br))

**Orientador:** Bernardo Lula Júnior ([lula@dsc.ufcg.edu.br](mailto:lula@dsc.ufcg.edu.br))

**Nível:** Mestrado

**Início:** Março 2006

**Conclusão(previsão):** Fevereiro 2008

**Resumo:** No estudo da computação quântica pode ser útil procurar traduzir conceitos da teoria da computação clássica para o contexto quântico. Alguns modelos de autômatos finitos quânticos têm sido introduzidos na literatura e parece interessante investigar suas aplicações práticas em problemas não satisfatoriamente resolvidos no contexto clássico. Neste artigo é apresentada uma maneira de se implementar, diretamente em um *hardware* quântico, um autômato finito não-determinístico com um custo proporcional ao tamanho da entrada.

**Palavras-chaves:** Autômatos finitos quânticos, Teoria da computação quântica.

# Um Algoritmo para Transformar Autômatos Finitos Não-Determinísticos em Autômatos Finitos Quânticos Preservando o Número de Estados e a Linguagem Reconhecida

Cheyenne R. G. Isidro, Bernardo Lula Júnior

Departamento de Sistemas e Computação – DSC  
Instituto de Estudos em Computação e Informação Quânticas – IQuanta  
Universidade Federal de Campina Grande - UFCG – Campina Grande, PB – Brasil  
{cha, lula}@dsc.ufcg.edu.br

**Abstract.** *In the study of quantum computation it can be useful to try to translate concepts of the theory of classical computation to the quantum context. Some quantum finitestate automaton models have been introduced in the literature and it seems interesting to investigate their practical application in problems that were not satisfactorily solved in the classical context. This article presents a way of implementing directly in quantum hardware a nondeterministic finitestate automaton with a proportional cost to the size of the input.*

**Resumo.** *No estudo da computação quântica pode ser útil procurar traduzir conceitos da teoria da computação clássica para o contexto quântico. Alguns modelos de autômatos finitos quânticos têm sido introduzidos na literatura e parece interessante investigar suas aplicações práticas em problemas não satisfatoriamente resolvidos no contexto clássico. Neste artigo é apresentada uma maneira de se implementar diretamente em um hardware quântico um autômato finito não-determinístico com um custo proporcional ao tamanho da entrada.*

## 1. Introdução

Autômatos finitos são elementos essenciais para o estudo da computação e constituem um modelo extremamente útil na elaboração de vários tipos diferentes de software: analisador léxico de um compilador; software para projetar e verificar o comportamento de circuitos digitais; software para examinar páginas da web a fim de encontrar ocorrências de palavras, frases ou outros padrões; software para verificar protocolos de comunicações, etc. [Hopcroft et al. 2001]. Formalmente, um *autômato finito* (AF) é uma 5-tupla  $A = (Q, \Sigma, \delta, q_{\text{init}}, Q_{\text{ac}})$ , onde  $Q$  é um conjunto finito de *estados*,  $\Sigma$  um *alfabeto de entrada*,  $\delta : Q \times \Sigma \rightarrow Q$  uma *função de transição*,  $q_{\text{init}} \in Q$  o *estado inicial* e  $Q_{\text{ac}} \subseteq Q$  um conjunto de *estados de aceitação*. O autômato  $A$  inicia sua computação no estado inicial  $q_{\text{init}}$  e lê uma palavra  $w = w_1w_2\dots w_n$  da esquerda para a direita, símbolo por símbolo. No  $i$ -ésimo passo, estando em um estado  $q$ ,  $A$  lê o símbolo  $w_i$  e atualiza seu estado para  $q' = \delta(q, w_i)$ .  $A$  *aceita* a palavra  $w$  se o estado no qual o autômato pára depois da leitura de todos os símbolos de  $w$  pertence a  $Q_{\text{ac}}$ .  $A$  *reconhece* a linguagem

formada pelas palavras aceitas. A classe de linguagens reconhecidas por autômatos finitos é a classe das linguagens *regulares*.

Esse tipo de autômato é dito ser *determinístico* (AFD), visto que o autômato está em apenas um estado em qualquer instante (o valor da função  $\delta$  é um elemento de  $Q$ ). Uma variante importante desse tipo de autômato é o autômato finito *não-determinístico* (AFN), que pode estar em vários estados ao mesmo tempo. Um AFN é essencialmente um AFD onde a função de transição  $\delta$  tem como valor um conjunto de estados em vez de apenas um estado, ou seja,  $\delta: Q \times \Sigma \rightarrow 2^Q$ . Um AFN aceita uma palavra  $w$  se alguma sequência de escolhas do próximo estado, enquanto são lidos os símbolos de  $w$ , levá-lo do estado inicial para algum estado de aceitação, ou seja, se existir algum *caminho de aceitação* dentre os vários caminhos possíveis. Essa extensão não-determinística, no entanto, não reconhece qualquer linguagem que não possa ser reconhecida por um autômato finito determinístico, ou seja, os dois conceitos AFD e AFN são equivalentes no que diz respeito à classe de linguagens reconhecidas. Apesar de não representar um ganho em termos de expansão da classe de linguagens reconhecidas, o uso de AFN's pode significar uma eficiência substancial tanto no tratamento formal (facilitar prova de alguns teoremas) quanto na descrição de uma aplicação. Na realidade, como diz Hopcroft em [Hopcroft et al. 2001]: "... o não-determinismo nos permite 'programar' soluções para problemas usando uma linguagem de alto-nível. O autômato finito não-determinístico é então 'compilado', por um algoritmo, em um autômato determinístico que pode ser então 'executado' em um computador convencional". Essa "compilação" é necessária e envolve a construção de todos os subconjuntos do conjunto de estados do AFN que serão então considerados como os estados do AFD equivalente. Embora esse número de estados possa ser minimizado, no pior caso, o menor AFD pode ter  $2^n$  estados enquanto o AFN original tem apenas  $n$  estados.

Este artigo apresenta um modelo de autômato finito quântico (AFQ) e um algoritmo de transformação de um AFN qualquer em um AFQ equivalente com o mesmo número de estados que aquele. A implementação de um AFQ construído conforme o modelo é mostrada através de um circuito quântico e o custo dessa implementação em termos de portas e qubit é avaliado.

## 2. Autômatos Finitos e Notação Vetorial

Uma outra formulação de autômatos finitos equivalente à utilizada acima e que é conveniente aos propósitos desse artigo é a formulação através de matrizes e vetores, conforme [Moore and Crutchfield 2000]. Se um autômato finito  $A$  (AFD ou AFN) tem  $n$  estados, sua função de transição  $\delta$  pode ser descrita por *matrizes de transição*  $\mathbf{M}_a$  de dimensão  $n \times n$ , uma para cada  $a \in \Sigma$ , de tal forma que  $(M_a)_{ij} = 1$  se e somente se  $A$  vai do estado  $q_i$  ao estado  $q_j$  ao ler o símbolo  $a$ , e  $(M_a)_{ij} = 0$ , caso essa transição não seja permitida. Então, se  $\mathbf{q}_{init}$  é o vetor coluna de um espaço de dimensão  $n$ :

$$(\mathbf{q}_{init})_i = \begin{cases} 1 & q_i = q_{init} \\ 0 & q_i \neq q_{init} \end{cases}$$

e  $\mathbf{Q}_{ac}$  é o vetor coluna:

$$(\mathbf{Q}_{ac})_i = \begin{cases} 1 & q_i \in Q_{ac} \\ 0 & q_i \notin Q_{ac} \end{cases}$$

o número de caminhos de aceitação para uma entrada  $w$  é dado pela função  $f$  definida como:

$$f(w) = \mathbf{q}_{init}^T \cdot \mathbf{M}_{w_1} \cdot \mathbf{M}_{w_2} \cdot \dots \cdot \mathbf{M}_{w_n} \cdot \mathbf{Q}_{ac} \quad (1)$$

(onde as operações são realizadas da esquerda para a direita).

Assim, uma palavra  $w$  é aceita se  $f(w) > 0$ , ou seja, se existe pelo menos um caminho levando do estado inicial a um estado de aceitação. Para fixar a notação utilizada, considere o seguinte exemplo de um autômato finito não-determinístico  $A$ , com  $Q = \{q_1, q_2\}$ ,  $\Sigma = \{0, 1\}$ ,  $q_{init} = q_1$ ,  $Q_{ac} = \{q_2\}$  e:

$$\delta(q_1, 0) = \{q_1, q_2\}, \delta(q_1, 1) = \{q_2\}, \delta(q_2, 0) = \{q_2\}, \delta(q_2, 1) = \{q_2\}.$$

Então, na notação matricial:

$$\mathbf{q}_{init} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{Q}_{ac} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \mathbf{M}_0 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \text{ e } \mathbf{M}_1 = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}$$

Ao computar a palavra  $w = 01$ :

$$f(01) = \mathbf{q}_{init}^T \cdot \mathbf{M}_0 \cdot \mathbf{M}_1 \cdot \mathbf{Q}_{ac} = \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 2$$

Ou seja, o valor de  $f(01) = 2$  indica que há dois caminhos que levam o autômato do estado inicial a um estado de aceitação ao ler a palavra  $w = 01$ , conforme árvore de derivação mostrada na figura 1 abaixo:

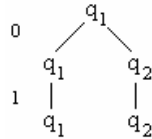


Figura 1. Árvore da computação de  $w = 01$  pelo AFN  $A$

### 3. Autômato Finito Quântico

Diversos modelos de autômatos finitos quânticos (AFQ) têm sido propostos ([Kondacs and Watrous 1997], [Moore and Crutchfield 2000], [Paschen 2000]). Basicamente, um AFQ, assim como um AF, é uma 5-tupla  $B = (Q, \Sigma, \delta, q_{init}, Q_{ac})$ , onde  $Q$  é um conjunto finito de estados,  $\Sigma$  é o alfabeto de entrada,  $\delta$  é a função de transição,  $q_{init}$  é o estado inicial e  $Q_{ac} \subseteq Q$  é o conjunto de estados de aceitação. A diferença para um AF é que em um AFQ os estados de  $Q$  podem estar em *superposição*. Uma superposição pode ser representada como um vetor unitário em um espaço de Hilbert de dimensão  $|Q|$ . Usando a base computacional e uma enumeração dos estados em  $Q$ , pode-se identificar o estado  $q_i$  com o  $i$ -ésimo vetor da base  $|q_i\rangle$  (usando a notação de Dirac) e representar uma superposição de estados como um vetor  $|q\rangle = \sum_{q_i \in Q} \alpha_i |q_i\rangle$ , onde  $\alpha_i$  é a *amplitude* (um

número complexo) do estado  $q_i$  e  $\sum_{i \in Q} |\alpha_i|^2 = 1$  (condição de normalização). A função de transição  $\delta$  é definida em termos de um mapeamento  $Q \times \Sigma \times Q \rightarrow \mathbb{C}$  e o valor de  $\delta(q_1, a, q_2)$  é a amplitude de  $|q_2\rangle$  na superposição de estados que B assume ao ler o símbolo  $a$  a partir de  $|q_1\rangle$ . Para  $a \in \Sigma$ ,  $U_a$  é uma matriz complexa de  $|Q| \times |Q|$  definida por:

$$U_a(|q_i\rangle) = \sum_{q' \in Q} \delta(q_i, a, q') |q'\rangle \quad (2)$$

Assim, ao ler um símbolo da entrada, um AFQ altera seu estado de superposição. Essa alteração deve preservar a condição de normalização, o que, em um espaço vetorial de dimensão finita significa que o correspondente operador deve ser *unitário*. Para isso, a seguinte condição deve ser satisfeita:

$$\forall q_i, q_j \in Q, a \in \Sigma : \sum_{q \in Q} \delta^*(q_i, a, q) \cdot \delta(q_j, a, q) = \begin{cases} 1 & \text{se } q_i = q_j \\ 0 & \text{caso contrário} \end{cases}$$

onde  $x^*$  para  $x \in \mathbb{C}$  é o *complexo conjugado* de  $x$ .

Um AFQ inicia sua computação no estado  $|q_{\text{init}}\rangle$  e ao ler cada símbolo  $w_i \in \Sigma$  da palavra de entrada  $w = w_1 w_2 \dots w_n$ , aplica a transformação unitária correspondente representada pela matriz unitária  $U_{w_i}$  ao estado do sistema representado pela superposição  $|q\rangle$ . Medições ortogonais (projeções) podem ser aplicadas para se determinar o estado do autômato. Os projetores (*observáveis*) são operadores que projetam o estado do autômato em subespaços ortogonais formados a partir dos conjuntos  $Q_{\text{ac}}$  e  $Q - Q_{\text{ac}}$ .

O modelo de AFQ proposto por Moore e Crutchfield [Moore and Crutchfield 2000] por exemplo, define uma única medição (MO) no final da computação da palavra  $w = w_1 w_2 \dots w_n$  através do projetor  $Q_{\text{ac}} = \text{span}\{|q\rangle : q \in Q_{\text{ac}}\}$ . Então, nesse modelo, usando a abreviação  $U_w$  para  $U_{w_n} \dots U_{w_2} U_{w_1}$ , a palavra  $w$  é aceita por um AFQ se e somente se,  $f_Q(w) = |Q_{\text{ac}} \cdot U_w \cdot |q_{\text{init}}\rangle|^2 > 0$ , ou seja, se ao final da computação a projeção deixa o autômato em um estado de aceitação (probabilidade maior que zero). A função  $f_Q$  para o caso quântico corresponde à função  $f$  definida pela equação (1) para o caso clássico.

O modelo de AFQ *1-way* introduzido por Kondacs e Watrous em [Kondacs and Watrous 1997] adiciona à formulação básica um sexto elemento, o conjunto  $Q_{\text{rej}}$  de estados de rejeição que, assim como os estados de aceitação, são considerados estados de parada (*halting states*) significando que o autômato pára ao atingir um desses estados, além de definir múltiplas medições (MM), uma a cada passo, através de um projetor nos subespaços ortogonais formados por  $\{|q\rangle : q \in Q_{\text{ac}}\}$ ,  $\{|q\rangle : q \in Q_{\text{rej}}\}$  e  $\{|q\rangle : q \in Q - (Q_{\text{ac}} \cup Q_{\text{rej}})\}$ . De acordo com [Ambainis and Freivalds 1998], qualquer linguagem reconhecida pelo modelo de Moore e Crutchfield é reconhecida pelo modelo

quântico de Kondacs e Watrous, mas a inversa não é verdadeira. As classes de linguagens reconhecidas por esses modelos são distintas e formam subconjuntos próprios da classe das linguagens regulares. Ou seja, são menos “poderosos” que o modelo clássico AF. Basicamente, a razão disso é a restrição de “unitariedade” das transformações  $U$ .

Um outro modelo de AFQ, AFQ *ancilla*, foi apresentado por Paschen em [Paschen 2000] que introduz qubits auxiliares de maneira a que as transformações sejam unitárias. Formalmente, um AFQ *ancilla*  $B$  é uma 6-tupla  $B = (Q, \Sigma, \Omega, \delta, q_{\text{init}}, Q_{\text{ac}})$ , onde  $Q, \Sigma, q_{\text{init}}$  e  $Q_{\text{ac}}$  são como no modelo de Moore e Crutchfield,  $\Omega$  é um alfabeto auxiliar e a função de transição é estendida para  $\delta: Q \times \Sigma \times Q \times \Omega \rightarrow C[0,1]$  com a restrição:

$$\forall q_i, q_j \in Q, a \in \Sigma: \sum_{y \in \Omega^*, q \in Q} \delta^*(q_i, a, q, y) \cdot \delta(q_j, a, q, y) = \begin{cases} 1 & \text{se } q_i = q_j \\ 0 & \text{caso contrário} \end{cases}$$

Com essa extensão, [Paschen 2000] mostra que toda linguagem regular pode ser reconhecida por um AFQ *ancilla* (lema 3.5).

#### 4. AFN e AFQ

Na prova do lema citado na seção anterior, Paschen caracteriza a linguagem reconhecida por um AFQ *ancilla* em termos do seu autômato mínimo (AFD com menor número de estados), ou seja, parte do autômato finito determinístico mínimo e mostra como construir um AFQ *ancilla* equivalente. Porém, apesar desse resultado, o procedimento utilizado não resolve o problema identificado na introdução desse artigo: como implementar diretamente um AFN sem passar pela transformação em AFD e incorrer em uma geração exponencial de estados. Uma solução para esse problema é apresentado a seguir através de um algoritmo para transformar diretamente um AFN em um AFQ *ancilla* com o mesmo número de estados e a mesma linguagem reconhecida que aquele.

**Algoritmo:** Seja  $A$  um AFN qualquer  $A = (Q, \Sigma, \delta, q_{\text{init}}, Q_{\text{ac}})$ . Definir um AFQ *ancilla*  $B = (Q', \Sigma, \Omega, \delta', q'_{\text{init}}, Q'_{\text{ac}})$  da seguinte forma:

1.  $Q' = \{|q\rangle : q \in Q\}$
2.  $\Omega = \{0,1\}$
3. Para cada entrada da função  $\delta(q_i, a) = \{q_{j_1}, \dots, q_{j_k}\}$  definir:

$$\delta'(q_i, a, q_{j_d}, 0) = \frac{1}{\sqrt{k}}, \text{ para todo } d, 1 \leq d \leq k, e$$

$$\delta'(q_i, a, q_s, 0) = 0, \text{ para todo } q_s \in Q - \{q_{j_1}, \dots, q_{j_k}\}$$

4.  $q'_{\text{init}} = |q_{\text{init}}\rangle$
5.  $Q'_{\text{ac}} = \text{span} \{|q\rangle : q \in Q_{\text{ac}}\}$
6. Para cada  $a \in \Sigma$ , definir a matriz  $U_a$  da seguinte forma:

$$U_a(|q_i\rangle|0\rangle) = |q_i\rangle \sum_{d=1}^k \delta'(q_i, a, q_{j_d}, 0) |q_{j_d}\rangle$$

com as outras entradas preenchidas com vetores ortonormais arbitrários de forma a que a matriz seja unitária ([Kondacs and Watrous 1997],[Ambainis and Freivalds 1998]).

**Proposição:**  $L(A) = L(B)$ .

**Prova** ( $L(A) \subseteq L(B)$ ): A prova é por indução no comprimento de  $w$ . Para simplificar, vamos considerar que  $U_a$  tem apenas um argumento como em (2),

$$U_a(|q_i\rangle) = \sum_{d=1}^k \delta'(q_i, a, q_{j_d}, 0) |q_{j_d}\rangle.$$

1. Seja  $w = a \in L(A)$ . Então,  $\delta(q_{init}, a) = \{q_{j_1}, \dots, q_{j_k}\}$ ,  $1 \leq k \leq |Q|$ , e  $\{q_{j_1}, \dots, q_{j_k}\} \cap Q_{ac} \neq \emptyset$ .

Por definição de  $B$ , se  $\delta(q_{init}, a) = \{q_{j_1}, \dots, q_{j_k}\}$ ,  $\delta'(q_{init}, a, q_{j_d}, 0) = \frac{1}{\sqrt{k}}$ , para todo  $d$ ,

$1 \leq d \leq k$ . Então,  $U_a(|q_{init}\rangle) = \sum_{d=1}^k \frac{1}{\sqrt{k}} |q_{j_d}\rangle$ . Como  $\{q_{j_1}, \dots, q_{j_k}\} \cap Q_{ac} \neq \emptyset$ , então

$$|Q'_{ac} U_a |q_{init}\rangle|^2 = \left| Q'_{ac} \sum_{d=1}^k \frac{1}{\sqrt{k}} |q_{j_d}\rangle \right|^2 > 0, \text{ o que implica em } a = w \in L(B).$$

2. Seja  $u = a_1 \dots a_n$  uma palavra qualquer de comprimento  $n$ . Então, por hipótese,

$\hat{\delta}(q_{init}, u) = \hat{\delta}(q_{init}, a_1 \dots a_n) = \{q_{j_1}, \dots, q_{j_m}\}$  se e somente se  $U_u(|q_{init}\rangle) = \sum_{d=1}^m \frac{1}{\sqrt{m}} |q_{j_d}\rangle$ , para

algum  $m$ , e onde  $\hat{\delta}$  é a extensão de  $\delta$  conforme [Hopcroft et al. 2001] e

$$U_u = U_{a_n} \dots U_{a_1}.$$

3. Seja  $w = ua \in L(A)$ ,  $|u| = n$  e  $a \in \Sigma$ . Como  $ua \in L(A)$ , então,

$$\hat{\delta}(q_{init}, ua) = \hat{\delta}(\hat{\delta}(q_{init}, u), a) = \hat{\delta}(\{q_{j_1}, \dots, q_{j_m}\}, a) = \bigcup_{d=1}^m \delta(q_{j_d}, a) \text{ e}$$

$$\bigcup_{d=1}^m \delta(q_{j_d}, a) \cap Q_{ac} \neq \emptyset.$$

Por hipótese,  $U_u(|q_{init}\rangle) = \sum_{d=1}^m \frac{1}{\sqrt{m}} |q_{j_d}\rangle$ , então  $U_a \left( \sum_{d=1}^m \frac{1}{\sqrt{m}} |q_{j_d}\rangle \right) = \frac{1}{\sqrt{m}} \sum_{d=1}^m U_a |q_{j_d}\rangle$ .

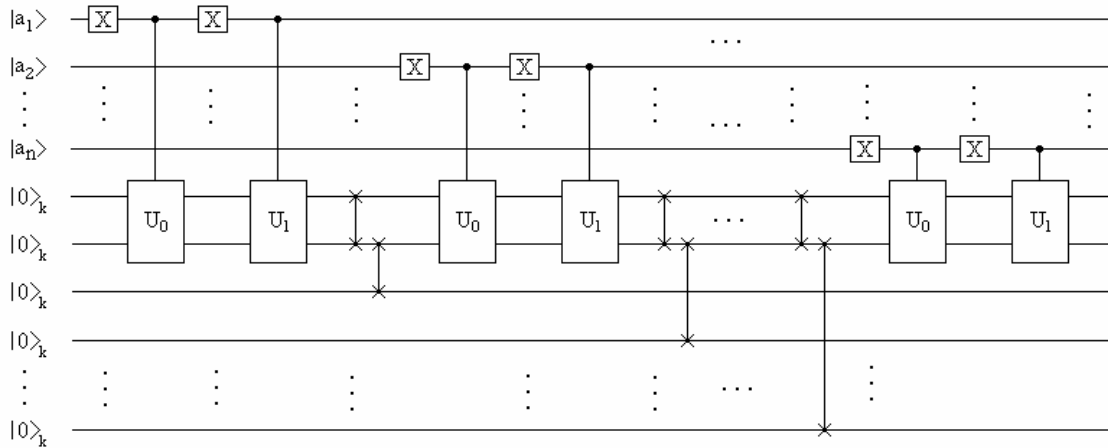
Como  $\bigcup_{d=1}^m \delta(q_{j_d}, a) \cap Q_{ac} \neq \emptyset$ , então  $\left| Q'_{ac} \left( \frac{1}{\sqrt{m}} \sum_{d=1}^m U_a |q_{j_d}\rangle \right) \right|^2 > 0$ , o que implica em

$ua = w \in L(B)$ . ■

A prova de  $L(B) \subseteq L(A)$  segue raciocínio similar.

## 5. Circuito Quântico para AFQ *ancilla*

Um AFQ *ancilla* B construído segundo o algoritmo descrito na seção anterior pode ser implementado com pouco esforço em um circuito quântico como mostra a figura 2 abaixo onde o alfabeto do autômato é  $\Sigma = \{0, 1\}$ .



**Figura 2. Circuito quântico que implementa um AFQ *ancilla* segundo o algoritmo.**

Esse circuito utiliza  $n$  qubits para representar uma palavra  $w \in \{0,1\}^n$ ,  $2k$  qubits para as duas entradas das portas  $U$  ( $k$  primeiros para representar o estado atual e  $k$  últimos para o próximo estado),  $k$  qubits extras no estado  $|0\rangle$  para cada símbolo da entrada e um grupo de 2 portas SWAP para “injetar” a segunda saída de uma porta  $U$  na primeira entrada da seguinte e restabelecer o estado  $|0\rangle$  na segunda entrada. Se a palavra de entrada  $w$  tem  $n$  símbolos, o número de qubits necessário é  $h = (k + 1) \cdot n + 2k$ , ou seja,  $h = O(n)$ . O número de portas no circuito é descrito por  $p = (2X + 2U) \cdot n + 2\text{SWAP} \cdot (n - 1)$ , ou seja,  $p = O(n)$ . Quanto às matrizes  $U_0$  e  $U_1$ , pode ser mostrado que é possível escrever essas matrizes definindo apenas  $|Q||\Omega|$  linhas e  $|Q|$  colunas [Paschen 2000].

## 6. Conclusão

Pode ser útil para um maior entendimento da computação quântica seguir os passos da teoria da computação clássica e tentar redefinir no contexto quântico alguns conceitos da teoria da computação clássica. De um ponto de vista prático, pode-se começar com os dispositivos mais simples da hierarquia computacional clássica, como os autômatos finitos, e tentar mostrar vantagens que podem ser obtidas com a utilização de um modelo quântico. Nesse artigo, foi mostrado que um autômato finito não-determinístico pode ser implementado diretamente em um *hardware* quântico sem incorrer em uma explosão de estados como previsto no caso clássico. O modelo de um AFQ com qubits auxiliares pode implementar diretamente um AFN através de um circuito quântico com custo que escala polinomialmente com o tamanho da entrada.

## **7. Agradecimentos**

O trabalho descrito neste artigo conta com financiamento parcial da CAPES através de uma bolsa de mestrado.

## **Referências**

Ambainis, A. and Freivalds, R. (1998) “1-Way Quantum Finite Automata: Strengths, Weaknesses and Generalizations”, FOCS 1998 Proceedings of the 39 Annual Symposium on Foundations of Computer Science: 332-341

Kondacs, A. and Watrous, J. (1997) “On the Power of Quantum Finite State Automata”, FOCS 1997: 66-75

Moore, C. and Crutchfield, J. P. (2000) “Quantum automata and quantum grammars”, In: Theor. Comput. Sci. 237(1-2): 275-306

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

Paschen, K. (2000) “Quantum finite automata using ancilla qubits”, University of Karlsruhe. Technical report.

Hopcroft, J. E., Motwani, R. and Ullman, J. D. (2001). Introduction to automata Theory, Languages and Computation. Addison-Wesley Publishing Company.