Revista Brasileira de Ensino de Física, v. 35, n. 1, 1306 (2013)
www.sbfisica.org.br

Introdução à programação quântica

(Introduction to quantum programming)

Marcelo Archanjo José, José Roberto Castilho Piqueira[1], Roseli de Deus Lopes

Escola Politécnica, Universidade de São Paulo, São Paulo, SP, Brasil
Recebido em 19/5/2012; Aceito em 23/6/2012; Publicado em 18/2/2013

Este trabalho apresenta uma visão geral sobre a programação quântica, de maneira acessível a alunos de graduação em Física. Inicia-se com um breve histórico sobre a computação quântica. São tratados os modelos que servem de base para a programação quântica, seguidos de descritivo resumido sobre as principais linguagens de programação quântica. Completando a apresentação, o funcionamento de um programa quântico é discutido, contextualizando-se a importância da programação, no âmbito da computação quântica.
Palavras-chave: programação quântica, computação quântica, teoria quântica da informação.

This work presents a general view about quantum programming, in an accessible way to undergraduate Physics students. It starts with a historical briefing about quantum computing. The basic models for quantum programming are treated, followed by the descriptions of the main quantum programming languages. Completing the presentation, it is discussed how a quantum program works, contextualizing how programming is important, concerning to quantum computation.
Keywords: quantum programming, quantum computation, quantum information theory.

1. Introdução

A programação de computadores é a base para explorar a capacidade disponibilizada pelo hardware. Quando se fala de computadores quânticos, logo se imagina como explorar suas capacidades, mas para isso é necessário programá-los.

No computador clássico, os programadores não precisam compreender os fenômenos físicos que permeiam o funcionamento do mesmo, mas precisam compreender a lógica de utilização dos seus recursos, como memória, operações e comunicação.

Diferentemente do computador clássico, no qual um bit pode assumir somente um de dois valores (0 e 1), no computador quântico, o qubit possui os dois valores (0 e 1) superpostos. Se o qubit for medido, seu valor irá colapsar para um dos dois valores. Sendo assim, não se pode medir o valor do qubit durante as operações, mas somente ao final dos processos.

Com o computador quântico, um programador também não precisa compreender os fenômenos da física quântica para poder utilizar todo o seu poder, mas precisa compreender a lógica desse novo paradigma da computação. Esta nova lógica possui três etapas fundamentais:

- preparação dos estados iniciais;
- realização das transformações unitárias;
- execução das medições.

A preparação dos estados iniciais pode ser entendida como a carga dos valores de entrada. Qualquer operação necessita de algum tipo de carga de dados e, uma vez que os estados iniciais estão preparados, as transformações unitárias realizam o verdadeiro processamento do computador quântico. Ao final de todas as transformações, pode ser executada a medição.

Essas três etapas são a base para a concepção dos computadores quânticos. Nielsen e Chuang [1] acrescentam que o computador quântico deve representar, de maneira robusta, a informação quântica. Essa característica traz uma grande dificuldade para a construção do computador quântico, um paradoxo entre duas restrições: o qubit deve ser suficientemente isolado para preservar a informação quântica e, ao mesmo tempo, deve permitir sua preparação e medição.

Nielsen e Chuang [1] enunciam: "uma implementação (do computador quântico) deve resolver esse delicado balanço entre as duas restrições. Sendo assim, a questão relevante não é como fazer o computador quântico, mas sim, quão bom o computador quântico pode ser feito".

Apesar dos computadores quânticos não estarem disponíveis para o público em geral, devido às dificuldades na sua implementação, avança-se no estudo de sua

[1]E-mail: piqueira@lac.usp.br.

programação. Para isso, são necessários modelos que representem seu funcionamento, servindo de base para a concepção das linguagens de programação, que visam explorar as potencialidades do computador quântico.

2. Histórico

Para entender a programação quântica, primeiramente é preciso conhecer como a mecânica quântica e a computação se relacionam e as suas consequências: a teoria quântica da informação e a computação quântica. Estes conceitos são encontrados em diversos artigos, [2-5].

Em 1982 Richard Feynman [6] e Paul Benioff [7], independentemente, apresentaram o conceito da computação quântica. Feynman foi motivado pela simulação de sistemas quânticos, que é computacionalmente dispendiosa em computadores clássicos, teorizando que um sistema quântico deve ser simulado mais eficiente e naturalmente em um computador quântico. Benioff, por sua vez, apontou que a miniaturização dos circuitos integrados já apresentava efeitos espúrios relacionados à mecânica quântica, afetando assim o comportamento dos circuitos. Ele argumentou que aproveitar intencionalmente desses efeitos da mecânica quântica ajudaria a superar a barreira de miniaturização de circuitos integrados e estabeleceria novos patamares na computação.

A ideia do computador quântico começou a tomar forma com David Deutsch em 1985 [8], que definiu a versão quântica da Máquina de Turing e de circuitos, demonstrando que o computador quântico seria capaz de realizar algumas tarefas de maneira mais eficiente que o computador clássico. Entretanto, os exemplos apresentados por ele, na ocasião, tinham aplicação limitada.

Quase dez anos depois, Peter Shor [9] em 1994, apresentou provavelmente o mais conhecido exemplo da eficiência do computador quântico, no qual são descritos algoritmos para logaritmos discretos e fatoração. Demonstrou que a fatoração em computadores quânticos seria realizada de maneira muito mais eficiente do que no computador clássico, devido ao paralelismo quântico. Depois desse artigo, o interesse pela computação quântica aumentou muito, a fatoração, custosa em computadores clássicos, é utilizada como base para muitos algoritmos criptográficos (mais detalhes sobre a criptografia RSA são apresentados no artigo de Rivest, Shamir e Adleman de 1978 [10]). A comunidade científica passou, desde então, a se esforçar ainda mais para desenvolver o computador quântico, mas ainda não há versões funcionais fora de ambientes de pesquisa.

3. Conceitos básicos

Essa nova abordagem computacional que a mecânica quântica apresenta não é intuitiva e, muitas vezes parece estranha. Por exemplo, na mecânica quântica, a polarização de um fóton, ao ser medida, pode colapsar para uma de duas diferentes polarizações (com probabilidades $|\alpha|^2$ e $|\beta|^2$, respectivamente) [1]. A polarização do fóton é considerada como um estado quântico, pois os dois possíveis resultados, de fato, convivem simultaneamente até ser realizada a medida. Esse fato é representado matematicamente por meio da notação de Dirac, utilizando $|x>$ e $|y>$ (lê-se ket x e ket y) como representativos das polarizações do fóton.

Isto é

$|\Psi>= \alpha|x> +\beta|y>$, sendo $|\Psi>$ o estado quântico do fóton.

Essa forma da mecânica quântica tratar uma solução binária, considerando a superposição, oferece uma nova abordagem na computação. Em vez de termos o bit estático (computador clássico) temos o qubit (bit quântico) como a superposição dos dois estados (0 e 1).

Essa também é a base do paralelismo quântico, uma operação com um qubit de saída possui intrinsecamente os dois resultados. É claro que isso pode fazer parte de um programa em um computador clássico, mas a diferença é que, no computador quântico, os dois estados já estão naturalmente superpostos, o que, em princípio, não parece muito importante, para um único qubit, porém quando a situação muda para múltiplos qubits, o benefício é exponencial.

Outra implicação é que α e β podem assumir quaisquer valores complexos, desde que: $|\alpha|^2+|\beta|^2 = 1$. Para ilustrar esse fato, pergunta-se: para duas variáveis de ponto flutuante, quantas informações estariam embutidas dentro de um único qubit?

Percebe-se, entretanto, que no computador quântico essas "variáveis" de fato não ocupam espaço de memória (bytes), pois α e β são intrínsecos ao próprio qubit. Por outro lado, quando ocorre a medição, o qubit irá colapsar para um dos estados. Isso é um problema, uma vez que o grande valor da superposição e do paralelismo quântico é praticamente anulado na medição.

A solução dessa questão [8, 11, 12] é realizar transformações unitárias que levem a uma seleção de resultados antes de realizar a medida. A ideia expressa nesse procedimento é explicável pelo fenômeno da interferência [13-16].

Para compreender como a interferência quântica funciona [17], é importante notar que os estados quânticos não são meramente distribuições de probabilidade de valores binários, mas sim vetores complexos. Dessa forma, podemos utilizar a interferência quântica para destacar estados de interesse ou para anular estados desinteressantes.

A implementação física do computador quântico apresenta, também, o entrelaçamento (*entanglement*). Segundo Ömer [18] o registrador quântico que contém mais de um qubit não pode ser descrito como simples-

mente uma lista de estados de cada qubit. De fato, esse registrador quântico (um conjunto de qubits) não é separável. Essa característica é bem diferente do que ocorre no computador clássico, nos quais 2 bits em um computador têm o mesmo comportamento de 1 bit em 2 computadores compondo a mesma informação (com o devido tratamento para isso).

Para esse mesmo exemplo, 2 qubits em um computador têm um comportamento diferente de 1 qubit em 2 computadores, tentando compor a mesma informação. Isso ocorre devido ao fato de que os qubits no mesmo computador quântico sofrem o efeito de entrelaçamento, isto é, o estado de um qubit interfere no estado do outro.

Dois qubits individuais são representados matematicamente da seguinte forma

$$|\Psi_1 >= a|0 > +b|1 >, \quad |a|^2 + |b|^2 = 1;$$
$$|\Psi_2 >= c|0 > +d|1 >, \quad |c|^2 + |d|^2 = 1.$$

Pode-se considerar que os qubits descritos por $|\Psi_1 >$ e $|\Psi_2 >$ estão localizados em computadores diferentes.

Dois qubits juntos são representados da seguinte forma

$$|\Psi_3 >= \text{e } |0,0 > +f|1,0 > +g|0,1 > +h|1,1 >,$$

$$\text{com } |\text{e}|^2 + |\text{f}|^2 + |\text{g}|^2 + |\text{h}|^2 = 1.$$

Esses dois qubits estão sob o efeito de entrelaçamento podendo ser considerarados como localizados no mesmo computador.

Por exemplo, se $|\text{h}|^2$ é muito pequeno (probabilidade de ocorrer $|1,1 >$), isto não define as outras probabilidades ($|\text{e}|^2$, $|\text{f}|^2$, $|\text{g}|^2$). Porém, se $|\text{b}|^2$ e $|\text{d}|^2$ também são pequenos (probabilidades de ocorrer $|1 >$ e $|1 >$) $|\text{a}|^2$ e $|\text{c}|^2$ deverão ser necessariamente grandes. Isto é, ao definir b e d são automaticamente definidos a e c. Sendo assim, fica claro que a, b, c, d não são respectivamente iguais e, f, g, h (para este exemplo), demonstrando que dois qubits no mesmo computador quântico não podem ser representados por dois qubits em dois computadores diferentes.

Outra característica importante na computação quântica é o teorema da impossibilidade de cópia (*no-cloning theorem*). Num computador clássico, um bit pode ser copiado, mas no computador quântico, isso não é possível. De uma forma simplificada, é necessário ter acesso ao dado para copiá-lo, mas, ao medir um qubit ele irá colapsar para um dos possíveis resultados, o que inviabiliza sua cópia. O enunciado e a demonstração detalhada da impossibilidade de cópia podem ser encontrados nas Refs. [19-23]

Um componente essencial dos computadores clássicos são as portas lógicas, "em um computador quântico o processo é similar, ou seja, pode-se construir (teoricamente) circuitos quânticos, que são agrupamentos de dispositivos mais simples chamados portas quânticas" [12]. Uma porta quântica que possui uma importância especial é a porta NOT-controlada (ou simplesmente CNOT), Fig. 1, pois ela permite a criação de estados quânticos entrelaçados, uteis nos programas quânticos [4, 24].

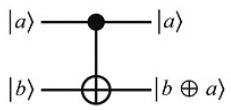

Figura 1 - Porta quântica CNOT, o circulo preto indica o qubit de controle e círculo vazado, o qubit alvo.

Enquanto não há computadores quânticos de uso geral, para avaliar seus potenciais pode-se conhecer e estudar a programação quântica e constatar a eficiência que determinados algoritmos teriam ao serem implementados em computadores quânticos [9, 25, 26].

Da mesma forma que antes da computação clássica ser estabelecida, foi possível simular seu funcionamento pelo modelo matemático da máquina de Turing, de maneira análoga, uma "máquina de Turing" quântica [8, 25] pode ser concebida. Isto é, uma simulação do que seria o comportamento de um computador quântico.

Nesse sentido, os computadores clássicos podem simular o computador quântico e, dessa forma, estabelecer o ambiente básico para o desenvolvimento de linguagens de programação e algoritmos próprios para o computador quântico, fomentando assim a programação quântica.

4. Programação quântica em computadores clássicos

A programação quântica em computadores clássicos é uma área bastante ativa e existe muita informação disponível a respeito. Três artigos relevantes são referência nos estudos dessa área: Sofge em 2008 [5], Gay em 2006 [2] e Selinger em 2004 [3]. Também a tese de Grattage em 2006 [4] apresenta descrições claras e objetivas.

A programação de computadores precisa ser realizada com o conhecimento da plataforma. Isto é, um programa clássico concebido para um processador Intel Pentium não funcionará diretamente em um processador ARM. Porém, é possível emular uma plataforma em outra. Por exemplo, é comum o desenvolvimento de programas para celulares utilizando PCs com emuladores correspondentes. É claro que o desempenho de um emulador tende a ser menor do que o próprio processador (com exceção do processador emulado ser muitas vezes inferior que o processador utilizado para emular). Especialmente no caso da emulação do computador quântico, três modelos de hardware virtual [3] que podem ser considerados equivalentes (pelo menos em teoria) merecem destaque:

- Modelo de Circuito Quântico – feito com portas quânticas da mesma maneira que um circuito lógico clássico. A diferença é que as portas quânticas, ao contrário das clássicas, são sempre reversíveis.
- Modelo da Máquina de Turing Quântica [8] – as medidas nunca são realizadas e a operação inteira da máquina é assumida como unitária.
- Modelo QRAM (*Quantum Random Access Machine*) [27] - diferentemente do modelo de circuito quântico, o modelo QRAM permite transformações unitárias e medidas serem livremente intercaladas. No modelo QRAM, um dispositivo quântico é controlado por um computador clássico. Esse dispositivo possui um enorme (mas finito) número de qubits individuais e endereçáveis, muito parecido com a memória RAM (*Random Access Memory*). O computador clássico envia uma sequência de instruções, às quais pode-se "aplicar uma transformação unitária U com os qubits i e j" ou "medir o qubit i". O dispositivo quântico executa essas instruções e disponibiliza os resultados. Esse modelo foi construído com a premissa de que o computador quântico real será, de fato, um computador clássico com acesso a componentes da computação quântica [5]. Diversas linguagens de programação quânticas usam o modelo QRAM.

As linguagens de programação quântica são classificadas pela literatura [2, 3, 5] como: linguagens imperativas, linguagens funcionais e outros paradigmas.

4.1. Linguagens imperativas

As linguagens imperativas também são conhecidas como linguagens procedurais. Elas são constituídas pelo uso de declarações que mudam o estado global do programa ou variáveis do sistema [5]. Exemplos de linguagens imperativas clássicas são FORTRAN, Pascal, C e Java. As linguagens imperativas são as mais utilizadas atualmente.

As linguagens imperativas de programação quântica são derivadas do trabalho de Knill [27], que, em 1996, apresentou as convenções para a escrita de algoritmos quânticos em pseudocódigo.

QCL de Bernhard Ömer [17, 18, 28, 29] foi a primeira linguagem de programação quântica real, apresentada em 1998, utilizando uma sintaxe derivada da linguagem C e provendo um simulador para o desenvolvimento e teste do código em um computador clássico. Suporta a visão do modelo de hardware virtual QRAM, no qual o computador clássico convive com o quântico (Fig. 2). Contém uma completa linguagem de programação clássica como uma sub-linguagem e provê um conjunto útil de funções de programação quântica de alto nível, como o gerenciamento de memória [2].

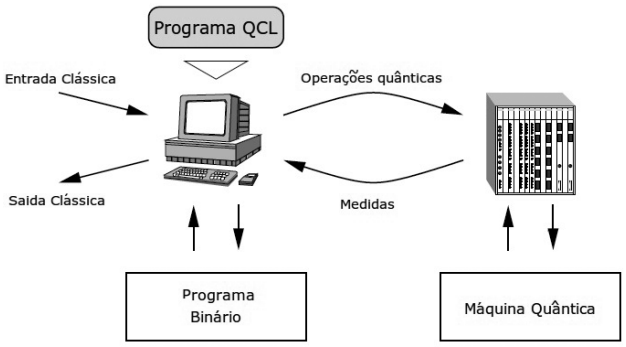

Figura 2 - Arquitetura quântica híbrida, figura extraída da Ref. [17], tradução dos autores.

Sanders e Zuliani [29], em 2000, e posteriormente Zuliani [30-32] apresentaram a qGCL, que é baseada na linguagem de comandos de Dijkstra. A qGCL é muito mais uma especificação de linguagem do que uma linguagem de programação.

Betteli, Calarco e Serafini [33], em 2003, apresentaram uma linguagem baseada em C++ chamada Q (existe outra linguagem Q – eQuation mais conhecida e não correlata), que possui classes importantes como QHadamard, QFourier, QNot, Qswap e Qop. Como recurso adicional, a linguagem possui parâmetros para a simulação de ruído.

Em 2004, Tafliovich [34] definiu uma linguagem de programação quântica que possui uma forte conexão entre os programas e as especificações. É focada na metodologia de programação, na qual o desenvolvimento dos programas é feito em pequenas etapas, e somente avança quando a etapa provar eficiência

Em 2007, Mlnarik [35] apresentou uma linguagem chamada LanQ que possui uma sintaxe similar à linguagem C, tratando tanto de operações clássicas como quânticas, oferecendo também ferramentas para a comunicação e execução de programas em paralelo.

4.2. Linguagens funcionais

Linguagens funcionais utilizam um tipo de paradigma para programação de computadores em que o foco das funções é puramente nas entradas e saídas, tal qual ocorre na Matemática, em oposição às linguagens imperativas que alteram estados e dados. APL e Lisp são exemplos de linguagens funcionais clássicas.

Maymin [36] apresentou, em 1996, um cálculo lambda com um foco probabilístico e outro quântico. Em 1997 [37], ele destaca que o cálculo lambda quântico pode simular de maneira eficiente um autômato celular quântico, particionado em uma dimensão.

Como Selinger descreve no artigo [3] de 2004, a linguagem QFC (*Quantum Flow Charts*) foi a primeira proposta de uma linguagem funcional quântica. Ela foi apresentada em detalhes no artigo [38], também em 2004. Na QFC, os programas são feitos por meio de uma versão funcional de *flowcharts*, mas também possui uma versão alternativa baseada em texto: a QPL

(*Quantum Programming Language*).

Tanto medidas quanto operações unitárias são realizadas diretamente pela linguagem de uma maneira que evita erros de escrita. Foram integradas com o mesmo formalismo, tanto as características clássicas quanto as quânticas. A linguagem é compilada dentro do modelo QRAM.

Van Tonder [39], em 2004, apresentou um cálculo lambda quântico, cuja linguagem é puramente quântica e não trata medidas.

Altenkirch e Grattage [4, 40, 41], em 2005, apresentaram uma linguagem chamada QML, que trata tanto controles quânticos quanto dados quânticos. A QML é baseada em lógica linear e, ao invés de realizar cópia dos estados quânticos (que não é possível, pela impossibilidade de cópia), realiza operações com portas CNOT, visando o compartilhamento. A QML também apresenta um *if* que analisa o dado quântico sem medi-lo.

Em 2011, Ying e Feng [42] demonstraram a relação entre programas quânticos escritos em linguagens de programação de baixo e alto nível. A linguagem *flow-chart* de baixo nível apresentada trata dados quânticos, mas os controles são clássicos, o que é compatível com o modelo QRAM.

Outros trabalhos relacionados à linguagem de programação quântica funcional, destacados por [5] foram apresentados por:

- Danos e cols. [43] em 2005 – Apresentaram estudo de um modelo de computação quântica não reversível que também apresenta notação para entrelaçamento, medidas e correções locais;
- Perdrix [44] em 2005 – Definiu um tipo de sistema que reflete o entrelaçamento de estados quântico, baseado em cálculo lambda;
- Mu e Bird [45] em 2001 – Programação quântica modelada em Haskell (mais informações em `http://www.haskell.org/`), por meio da definição de um tipo de dado para registradores quânticos.
- Sabry em [46] 2003 – Sabry ampliou o modelo de Mu e Brid incluindo a representação de estados em entrelaçamento. Outros esforços relacionados a esta linha foram realizados por Vizzotto e Da Rocha Costa [47] em 2005, Karczmarczuk [48] em 2003 e Skibinski [49] em 2001.

4.3. Outros paradigmas de programação

Segundo Sofge [5], há outras abordagens que agem de uma forma bastante diferente no sentido de realizar a programação quântica. Freedman, Kitaev e Wong [50], em 2000, apresentaram a simulação de *topological quantum field theories* (TQFT's) para computação quântica. TQFT é um modelo mais robusto na computação quântica, por representar estados quânticos como sistemas físicos resistentes a perturbações.

Operações quânticas são determinadas a partir de propriedades topológicas globais, como por exemplo, os caminhos seguidos por partículas. Essa abordagem, bem diferente para a computação quântica pode promover novas ideias e fomentar a criação de novos algoritmos quânticos.

Em contrapartida, da maneira que está formulada, somente trata a evolução do estado e não trata o processo de medida. Atualmente, vários trabalhos foram desenvolvidos com o intuito de definir linguagens para suportar protocolos de criptografia quântica e, especialmente, tratar a inclusão da comunicação entre processos quânticos, sendo que esses processos podem ser locais ou remotos, originando a especificação da programação quântica distribuída. Os trabalhos que seguem merecem destaque nesta linha:

- Mauerer [51] em 2005 cQPL – Linguagen baseada na QPL, mas com extensões envolvendo comunicação entre processos distribuídos;
- Jorrand e Lalire [52] em 2005 QPAlg (*Quantum Process Algebra*) – Descreve interações entre processos quânticos e clássicos;
- Gay e Nagarajan [53] em 2005 CQP (*Communicating Quantum Processes*) – Linguagem para modelagem da comunicação de sistemas clássicos e quânticos com enfase na criptografia quântica;
- Adão e Mateus [54] em 2005 – Desenvolveram o processo de cálculo para protocolos seguros feito sobre o modelo computacional QRAM;
- Udrescu, Prodan e Vlâdutiu [55] em 2004 – Criaram uma linguagem descritiva de hardware para o desenvolvimento de circuitos quânticos (similar ao VLSI).

A Tabela 1 apresenta de maneira concisa as principais linguagens de programação quântica.

5. Exemplo de programa quântico para execução em computadores clássicos

É importante apresentar as características de um programa quântico. Para isso, foi escolhida a linguagem QCL, que é uma linguagem imperativa (procedural) porque hoje (nos computadores clássicos), as principais linguagens de programação são imperativas, como C/C++, Java. É claro que não há garantias de que este paradigma será o mais eficiente no computador quântico, mas ao considerarmos o modelo QRAM, que pressupõe a convivência do computador clássico e do quântico, há uma indicação lógica da importância das linguagens imperativas. Nessa linha, a linguagem QCL é uma referência na literatura, inclusive para novas linguagens como a LanQ.

Um programa em QCL segue o tipo de fluxo apresentado na Fig. 3.

Tabela 1 - Listagem das linguagens de programação quântica com suas referências.

Ano início	Autor / Referências	Nome da linguagem
	Linguagens imperativas	
1998	Ömer [17, 18, 28]	QCL
2000	Sanders e Zuliani [29] Zuliani [30-32]	qGCL
2003	Betteli, Calarco e Serafini [33]	Q[1]
2004	Tafliovich [34]	-
2007	Mlnarik [35]	LanQ
	Linguagens funcionais	
1996	Maymin [36]	-
2001	Mu eBird [45]	-
2001	Skibinski [49]	-
2003	Sabry [46]	-
2003	Karczmarczuk [48]	-
2004	Selinger [3, 8]	QFC e QPL
2004	Van Tonder [39]	-
2005	Altenkirch e Grattage [40, 41]	QML
2005	Danos, Kashefi e Panangaden [43]	-
2005	Perdrix, [44]	-
2005	Vizzotto e Da Rocha Costa [47]	-
2011	Ying e Feng [42]	-
	Outros paradigmas de programação	
2004	Udrescu, Prodan e Vlâdutiu [55]	-
2005	Mauerer [51]	cQPL
2005	Jorrand e Lalire [52]	QPAlg
2005	Gay e Nagarajan, [53]	CQP
2005	Adão e Mateus [54]	-

[1]Existe outra linguagem Q (eQuation) que não é relacionada a esta linguagem quântica.

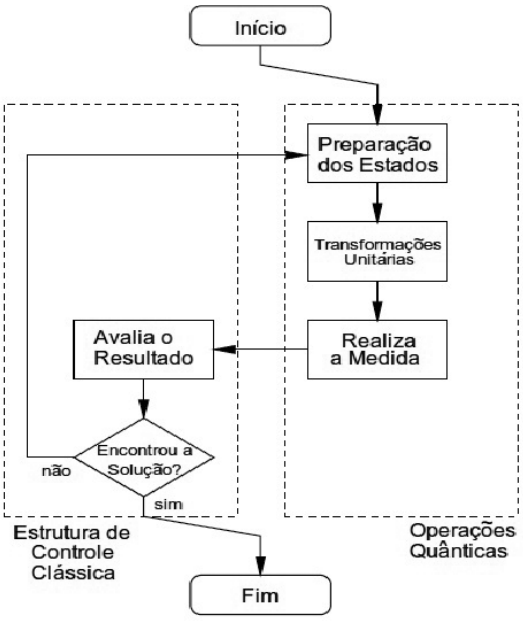

Figura 3 - Fluxo de um algoritmo quântico probabilístico, figura extraída da Ref. [17], tradução dos autores.

O algoritmo de Deutsch, descrito detalhadamente na Ref. [8], implementado em QCL, está descrito abaixo (deutsch.qcl):

```
/* Define Oracle */
const coin1=(random()¿=0.5); // Define two random boolean
const coin2=(random()¿=0.5); // constants
boolean g(boolean x) { // Oracle function g
if coin1 { // coin1=true -¿ g is constant
return coin2;
} else { // coin1=false -¿ g is balanced
return x xor coin2;
}
}
qufunct G(quconst x,quvoid y) { // Construct oracle op. G from g
if g(false) xor g(true) { CNot(y,x); }
if g(false) { Not(y); }
}
/* Deutsch's Algorithm */
operator U(qureg x,qureg y) { // Bundle all unitary operations
H(x); // of the algorithm into one
G(x,y); // operator U
H(x & y);
}
procedure deutsch() { // Classical control structure
qureg x[1]; // allocate 2 qubits
qureg y[1];
int m;
{ // evaluation loop
reset; // initialize machine state
U(x,y); // do unitary computation
measure y,m; // measure 2nd register
```

```
} until m==1; // value in 1st register valid?
measure x,m; // measure 1st register which
print "g(0) xor g(1) =",m; // contains g(0) xor g(1)
reset; // clean up
}
```

Para analisar o código acima, deve-se ter em conta os seguintes pontos:

// : trata-se de comentário: o texto à direita das duas barras não tem efeito no funcionamento do código;

/ */*: trata-se também de comentário; o texto entre */** e **/* não tem efeito no funcionamento do código.

Para explicar o funcionamento do programa quântico, realizar-se-á uma comparação com a descrição matemática do algoritmo de Deutsch [8] descrito por Ömer [17]:

1. O ponto inicial do programa é na *procedure deutsch()*

2 Preparação do estado inicial vazio

$|\psi_0> = \ |0>_x|0>_y$
Alocação do qubit x: *qureg x[1];*
Alocação do qubit y: *qureg y[1];*
Preparação dos estados iniciais vazios: *reset;*

3 Aplica-se a transformação de Hadamard H(x)

$$|\psi_1> = (H|0>_x)|0>_y = \frac{1}{\sqrt{2}}\left(|0>_x|0>_y + |1>_x|0>_y\right)$$

Dentro da sequência de execução quântica do programa, é chamado o operador $U(x, y)$;que é responsável pelas operações unitárias, sendo que a primeira delas é $H(x)$; aplicando a transformação de Hadamard no qubit x.

4 Aplica o operador Oráculo G

$|\psi_2> = \frac{1}{\sqrt{2}}\left(|0>_x|g(0)>_y + |1>_x|g(1)>_y\right)$
Ainda dentro do operador $U(x, y)$**;** há a chamada da função quântica $G(x, y)$;

5 Aplica H(x o y), resultando em

$$|\psi_3> = \frac{1}{2\sqrt{2}}\sum_{x\in B}\sum_{y\in B}(-1)^{yg(0)} + \left((-1)^{x+yg(1)}\right)|x>_x|y>_y.$$

Ainda dentro do operador $U(x, y)$; obtém-se *H(x & y);*

6 Uma vez terminado o processamento do operador $U(x, y)$; a execução do programa volta para a *procedure deutsch()* na qual é realizada a Medida de "x" e "y" pela ação de *measure y,m;* e *measure x,m;*

Para chamar a *procedure deutsch()* é necessário criar um programa simples, para isso cria-se um arquivo chamado my.qcl com o seguinte conteúdo:
include "deutsch";
deutsch();

E para que este programa apresente seus resultados, a partir do Terminal (no diretório onde está my.qcl), deve-se executar:
qcl my.qcl

O resultado apresentado pode ser, aleatoriamente, um destes:
QCL Quantum Computation Language (64 qubits, seed 1289349674)
: g(0) xor g(1) = 0
QCL Quantum Computation Language (64 qubits, seed 1289349831)
: g(0) xor g(1) = 1

"0" e "1" são os dois possíveis resultados do algoritmo de Deutsch.

6. Instalação do QCL

Para a execução de qualquer o programa na linguagem QCL, é preciso primeiramente instalar os binários do QCL. Sua distribuição pode ser encontrada em `http://tph.tuwien.ac.at/~oemer/qcl.html`.

Especificamente, no caso aqui descrito, foi utilizada a Binary Distribution (64 bit): qcl-0.6.3-x86_64-linux (AMD64, Linux 2.6, glibc2.4) que foi instalada em um Linux Ubuntu 10.04 de 64 bits. Caso seja necessário repetir este processo, seguem algumas recomendações importantes:

- ler o arquivo README da distribuição do QCL que for utilizar;
- especificamente no Linux Ubuntu 10.04 de 64 bits, instalar, por meio do Synaptics, os seguintes pacotes, antes de instalar o QCL:
  - libplot
  - flex
  - bison
  - readline5

Após a instalação com sucesso do QCL, já é possível executar os exemplos contidos no diretório:
/usr/local/lib/qcl/

Para executar um teste da instalação, basta executar o comando:
qcl test.qcl

Aparecendo resultados semelhantes aos seguintes:

```
QCL Quantum Computation Language (64 qubits, seed 1289349171)
shor(15);
: chosen random x = 11
: measured zero in 1st register. trying again ...
: chosen random x = 11
: measured 128 , approximation for 0.5 is 1 / 2
: possible period is 2
: 11  1 + 1 mod 15 = 12 , 11  1 - 1 mod 15 = 10
: 15 = 5 * 3
grover(123);
: 7 qubits, using 5 iterations
: measured 123
testexpn();
: testing modular exponentiation (7 x mod 31)
testdft();
: testing discrete Fourier transform
: expecting (1+i)/2 |000000 > - i/2 |010000 > + 1/2 |110000 >
: STATE: 6 / 64 qubits allocated, 58 / 64 qubits free
(0.5+0.5i) |0 > - 0.5i |16 > + 0.5 |48 >
```

7. Conclusões

Conceitos da programação quântica foram apresentados, especialmente ilustrando como um programa quântico pode ser emulado em um computador clássico. Este é um campo bastante amplo e com muita informação. Entre as referências que foram utilizadas, algumas merecem um destaque especial. Ömer [17] tem um trabalho bastante completo e apresenta descrições muito didáticas sobre questões importantes da programação quântica. Cabral e cols. [12] apresentam o algoritmo de Deutsch de maneira bastante didática.

É importante ter em mente que a mecânica quântica apresenta conceitos não intuitivos e surpreendentes, mas é necessário compreender que a utilização desses conceitos nas diversas áreas do conhecimento pode trazer benefícios igualmente surpreendentes.

Referências

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

[2] S. J. Gay, Mathematical Structures in Computer Science, **16**, 581 (2006).

[3] P. Selinger, in: *Proceedings of the 7th International Symposium on Functional and Logic Programming*, v. 2998 of Lecture Notes in Computer Science (Springer, Berlin, 2004a).

[4] J.J. Grattage, *A Functional Quantum Programming Language*. PhD thesis, The University of Nottingham 2006, disponível em `http://etheses.nottingham.ac.uk/250/1/thesis.pdf`, consultado em 14/10/2012.

[5] D.A. Sofge, in *Second International Conference on Quantum, Nano and Micro Technologies* (IEEE Computer Society Press, Sainte Luce, Martinique 2008).

[6] R.P. Feynman, International Journal of Theoretical Physics **21**, 467 (1982).

[7] P. Benioff, Physics Review Letters **48**, 1581 (1982).

[8] D. Deutsch, Proceedings of the Royal Society of London A **400**, 97 (1985).

[9] P. Shor, in *Proceedings of the 35th IEEE FOCS* (IEEE Computer Society Press, Santa Fé, México, 1994), p. 124-134, disponível em `http://ieeexplore.ieee.org/xpl/tocresult.jsp?isnumber=8384`, consultado em 14/10/2012.

[10] R.L. Rivest, Communications of the ACM **21**, 120 (1978).

[11] D. Deutsch and R. Jozsa, Proc. Roy. Soc. London, Ser. A **439**, 553 (1992).

[12] G.E.M. Cabral, A.F. Lima and B. Lula Jr.. Junior, Revista Brasileira de Ensino de Física **26**, 109 (2004).

[13] O. Pessoa Jr., Cadernos de História e Filosofia da Ciência Série 3 **2**, 177 (1992).

[14] O. Pessoa Jr., Revista Brasileira de Ensino de Física **19**, 27 (1997).

[15] O. Pessoa Jr., *Conceitos de Física Quântica* (Livraria da Física, São Paulo, 2003).

[16] F. Ostermann, Revista Brasileira de Ensino de Física **27**, 193 (2005).

[17] B. Ömer, *Structured Quantum Programming*. PhD thesis, Technical University of Vienna, 2003.

[18] B. Ömer, *A Procedural Formalism for Quantum Computing*. Master thesis, Technical University of Vienna, 1998.

[19] J.R.C. Piqueira, Revista Brasileira de Ensino de Física **33**, 4303 (2011).

[20] M. Hirvensalo, *Quantum Computing* (Springer, Berlin, 2001).

[21] G. Jaeger, *Quantum Information: An Overview* (Springer, New York, 2007).

[22] V. Vedral, *Introduction to Quantum Information* (Oxford University Press, Oxford, 2006).

[23] E. Desurvire, *Classical and Quantum Information Theory: An Introduction for the Telecom Scientist* (Cambridge University Press, Cambridge, 2009).

[24] P. Arrighi and G. Dowek, *Proceedings of the International Workshop on Quantum Programming Languages* (IEEE Computer Society Press, Turku, Finland, 2004), p. 21–38.

[25] C.A. Lungarzo, Cienc. Cult. **55**, 4 (2003).

[26] W.V. Dam and G. Seroussi, Report Number: HPL-2002-208 (University of California at Berkley, 2002).

[27] E. Knill, Technical Report LAUR-96-2724 (Los Alamos National Laboratory, Los Alamos, 1996).

[28] B. Ömer, *Quantum Programming in QCL*. Master's thesis, Technical University of Vienna, 2000.

[29] J.W.Sanders and P. Zuliani, in *Mathematics of Program Construction*, volume 1837 of Lecture Notes in Computer Science (Springer, Berlin, 2000).

[30] P. Zuliani, IBM J. Research and Development **45**, 807 (2001).

[31] P. Zuliani, in *Proceedings of the 2nd International Workshop on Quantum Programming Languages*, (IEEE Computer Society Press, Turku, Finland, 2004), p. 179-195.

[32] P. Zuliani, in *Proceedings of the 3rd International Workshop on Quantum Programming Languages* (IEEE Computer Society Press, Chicago, Illinois, 2005), p. 169-179.

[33] S. Bettelli, T. Calarco and L. Serafini, The European Physical Journal D **25**, 181 (2003).

[34] A. Tafliovich, *Quantum Programming*. Master's thesis, University of Toronto, 2004.

[35] H. Mlnarik, Operational semantics and type soundness of quantum programming language LanQ (arXiv:quant-ph/0708.0890v1, 2007).

[36] P. Maymin, Extending the lambda calculus to express randomized and quantumized algorithms (arXiv:quant-ph/9612052,1996).

[37] P. Maymin, The lambda-q calculus can efficiently simulate quantum computers (arXiv:quant-ph/9702057, 1997).

[38] P. Selinger, *Mathematical Structures in Computer Science*, **14**, 527 (2004b).

[39] A. Van Tonder, SIAM Journal on Computing **33**, 1109 (2004).

[40] T. Altenkirch and J. Grattage, in *Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science* (IEEE Computer Society, Washington, D.C., 2005a).

[41] T.Altenkirch and J. Grattage, *QML: Quantum Data and Control* `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.113.9509`, 2005b.

[42] M. Ying and Y. Feng, IEEE Transactions on Software Engineering **37**, 485 (2011).

[43] V. Danos, E. Kashefi and P. Panangaden, Physical Review A **72**, 064301 (2005).

[44] 4 S. Perdrix, in *Proceedings of the 3rd International Workshop on Quantum Programming Languages*, Electronic Notes in Theoretical Computer Science (Elsevier Science, Chicago, Illinois, 2005).

[45] S. Mu and R. Bird, in *Proceedings of the 2nd Asian Workshop on Programming Languages and Systems* (Korea Advanced Institute of Science and Technology, Daejeo, Korea, 2001).

[46] A. Sabry, in *Proceedings of the ACM SIGPLAN Workshop on Haskell* (ACM Press, Bloomington, Indiana, 2003).

[47] J.K. Vizzotto and A.C. da Rocha Costa, in *VII Congresso Brasileiro de Redes Neurais*, Sessão de Computação Quântica (Sociedade Brasileira de Redes Neurais, Natal-RN, 2005).

[48] J. Karczmarczuk, in *Proceedings of the ACM SIGPLAN*, Workshop on Haskell (ACM Press, Bloomington, Indiana, 2003).

[49] J. Skibinski, *Haskell Simulator of Quantum Computer*, disponível em `http://web.archive.org/web/20010520121707/www.numeric-quest.com/haskell`.

[50] M. Freedman, A. Kitaev and Z. Wong, arXiv:quant-ph/0001071/v3, 2000.

[51] W. Mauerer, *Semantics and Simulation of Communication in Quantum Computing*. Master's thesis, University Erlangen-Nuremberg, 2005.

[52] P. Jorrand and M. Lalire, *Unconventional Programming Paradigms*, LNCS Series, v. 3566 (Springer, Le Mont Saint Michel, França, 2005).

[53] S. Gay and R. Nagarajan, in *Proceedings of the 32nd ACM Annual Symposium on Principles of Programming Languages*, (ACM Press, New York, USA, 2005), p. 145-157.

[54] P. Adão and P. Mateus, in *Proceedings of the 3rd International Workshop on Quantum Programming Languages*, Electronic Notes in Theoretical Computer Science (Elsevier Science, Chicago, Illinois, 2005).

[55] M. Udrescu, L. Prodan, and M. Vlâdutiu, in *Proceedings of the 1st ACM Conference on Computing Frontiers* (ACM Press, Ischia, Italy, 2004).