Annales UMCS Informatica AI 2 (2004) 453-461

Annales UMCS
Informatica
Lublin-Polonia
Sectio AI

http://www.annales.umcs.lublin.pl/

Didactic tools for teaching quantum informatics

Piotr Gawron*, Jarosław Miszczak

The Institute of Theoretical and Applied Informatics of the Polish Academy of Sciences, Bałtycka 5, 44-100 Gliwice, Poland

Abstract

Development of quantum informatics as a new field of computer science poses new challenges to teachers and students of computer science. Among others, very dynamic branches of quantum informatics are quantum programming languages and simulation of quantum systems.

This article presents the program of the lecture: "Quantum systems of informatics" that is proposed to students of last semesters of computer science at Faculty of Automatic Control, Electronics and Computer Science of Silesian University of Technology

The objective of this lecture is to familiarize students with notions and elements of quantum informatics. Lecture will introduce basic mathematical concepts needed to operate a quantum mechanics apparatus and elements of information theory.

Many tools useful in education of quantum informatics are presented. Most of them are based on Quantum Computer Language. They are the authors' original work.

In the last section the examples of programs of laboratory classes, during which described software is used, are presented.

1. Introduction

Achievements made in the field of quantum informatics during the last 20 years imply strong need for education of quantum computation theory. Despite being knowledgeable one cannot use real quantum computer to perform useful computation, there exist some ready to use quantum systems of informatics offered commercially. The best example is the quantum key distribution (QKD) hardware [1-3], that is able to securely exchange cryptographic key at the distance of more than hundred kilometers. Students of computer engineering and computer science should be prepared for a new, quantum aspect of computer science. In this article we present the sketch of lecture: "Quantum systems of informatics" and presentation of software tools useful in simulation and visualization of quantum algorithms.

* Corresponding author: *e-mail address*: {gawron, miszczak}@iitis.gliwice.pl

2. Lecture

It seems obvious that students of computer science should be interested in learning its new aspects especially that development of quantum computers will have a very strong impact on security of computer systems.

We propose to divide our course into three general parts:

1. Elements of linear algebra and information theory,
   - Revision of such mathematical notions as: finite dimensional complex vector space, scalar product, tensor product, unitary operations and projection operator.
   - Measures of information and notion of entropy.

Basic notions of quantum informatics.

- Introduction to quantum mechanics.
- Quantum Turing machine [4].
- One and many qubit systems.
- Quantum elementary and complex gates.
- Reversible computation.
  - Landauer principle [5],
  - methods of generation of quantum and classical reversible circuits.
- Quantum algorithms: Deutsch's, Grover's search algorithm, quantum Fourier transform, Simon's and Shor's algorithms.
  - Physical realizations [6, 7].
- Quantum information theory:
  - Quantum teleportation.
  - Dense coding.
- Quantum cryptography.
  - Quantum Key Distribution [1, 2],
  - Methods of attack on contemporary cryptographic systems with the use of quantum computer.

3. Quantum programming languages, simulation and applications of quantum systems.
   - Simulation of quantum machines: on the gate level, on the circuit level.
   - Quantum programming languages: QCL, libquantum.

Program of this lecture is wide enough to discuss the key aspects of quantum informatics, without going into unnecessary details. Its main goal is to familiarize students with a completely new approach to computer science.

3. Description of tools

Quantum Computer Language [8-10] is created by Berhnard Oemer and it is the first realization of the language dedicated to quantum computing. From the user point of view QCL resembles classical procedural language like C or Pascal, but contains the elements needed for quantum programming in the

example quantum variables and quantum operators. QCL is designed for programming computer based on hybrid, quantum-classical architecture [11], which means that most of instructions run on classical computer and quantum device is used only to perform purely quantum operations.

Full documentation of QCL can be found in [8-10].

As an example in Table 1 we present the code of quantum full adder written in QCL.

Table 1. Implementation of full adder in QCL

```
operator carry(qureg cin, qureg a, qureg b, quvoid cout ) { // cin –
carry bit from previous level
    CNot(cout, a&b); // a – bit to add
    CNot(b, a); // b – bit to be added
    CNot(cout, cin&b); // cout – carry bit to next level
}

operator sum(qureg c, qureg a, qureg b) {   // c – carry bit from
previous level
    CNot(b, a);      // a – bit to add
    CNot(b, c);      // b – bit to be added
}

operator add(qureg a, qureg b) {// output: a:=a+b
  qureg s[#a];// temporary register
  int i;
  int n=#a;
  for i=0 to n-2 {
    carry(s[i], a[i], b[i], s[i+1]);// test all caries
  }
  carry(s[n-1], a[n-1], b[n-1], b[n]);
  CNot(b[n-1], a[n-1]);
  sum(s[n-1], a[n-1], b[n-1]);
  if (n>=2) {
    for i=n-2 to 0 step -1 {
      !carry(s[i], a[i], b[i], s[i+1]);// uncompute carries
      sum(s[i], a[i], b[i]);// perform sum for each bit
    }
  }
}
```

Examining this code one can notice the mixture of classical programming structures as “for-to-step”, and “if” with the quantum primitives: unitary “operator” and quantum gate “CNot”.

High level QCL program is translated into quantum circuit composed of elementary gates. QCL interpreter is integrated with qlib simulation library which substitute quantum computer by its simulator.

Procedural approach in programming of quantum computers makes programs scalable and easier to create and understand. Although one cannot know if QCL will be used to program real quantum computers it is a very powerful tool for quantum algorithms research and simulation.

Quantum Computer Simulator Interface is GUI program designed to educate the basics of quantum algorithms. It gives the user capability of interactive composing quantum circuit and observing change of amplitudes during evolution of quantum register. Basic quantum gates: not, controlled not, hadamard, rotation, controlled phase shift are available to the user. Those gates are sufficient to compose any quantum algorithm. Full description of program can be found in [12].

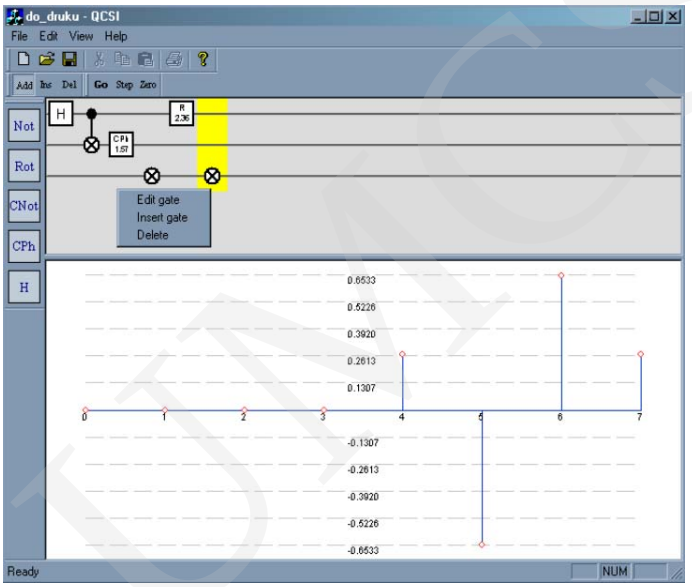


Fig. 1. QCSI main window

QCL2eps is the java program that translates description of quantum circuit into its graphical representation. This program uses output of QCL interpreter, parses it and generates output in the form of postscript graphic. It can be very useful when debugging QCL programs. QCL2eps project is still in development. The example of its capabilities is shown below.

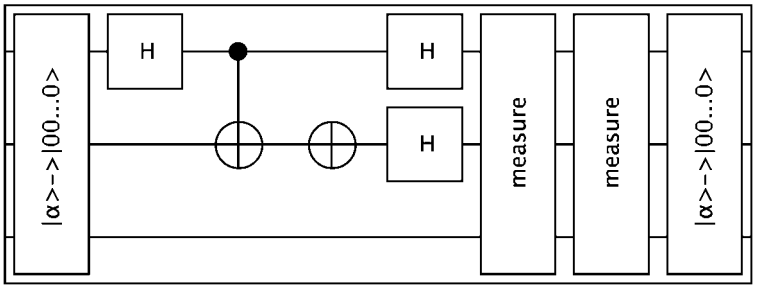


Fig. 2. Deutsch's algorithm drawn by the QCL2eps program

Quantum-octave [13] is a set of functions written in Gnu Octave language [14] giving possibility to simulate quantum systems. What is different in this simulation package [15,16] is that quantum-octave operates on density operators being able to represent mixed states. Documentation of package can be found on the project web page [13].

The example in Table 2 presents simulation of Grover's algorithm in two cases. Before application of algorithm quantum register is prepared in a pure state or in some mixed state. Probability distributions of measured outcomes are presented in Fig. 3.

Table 2. Example of usage of quantum-octave

```
function demo()
pure_state0 = Ket([0,0,0]); #prepare some pure sates
pure_state1 = Ket([1,0,0]);
pure_state2 = Ket([0,1,0]);
pure_state6 = Ket([0,1,1]);

mixed_state0 = State(pure_state0); #make mixed states from pure states
mixed_state1 = State(pure_state1);
mixed_state2 = State(pure_state2);
mixed_state6 = State(pure_state6);

PlotProbs(MeasureZ(State(pure_state0))); # plot pure state |000>; prob.
1.

# make Grover's algorithm beginning with state |000>
grover_state_pure = grover(5, State(pure_state0));
outcome_post_pure = MeasureZ(grover_state_pure);

PlotProbs(outcome_post_pure); # see how does it look; prob. 2.

mixed_state   =   MixStates(20,   mixed_state0,   1,   mixed_state1,   2,
mixed_state2, 3, mixed_state6);
# make statistical mixture of states

outcome_pre_mixed = MeasureZ(mixed_state); # mesasure this mixture

PlotProbs(outcome_pre_mixed); # plot probabilites of this mixture; prob.
3.

# make Grover's algorithm begining in our mixted state
grover_state_mixed = grover(5, mixed_state);
outcome_post_mixed = MeasureZ(grover_state_mixed);

PlotProbs(outcome_post_mixed); # see how does look its outcome; prob. 4.
endfunction
```

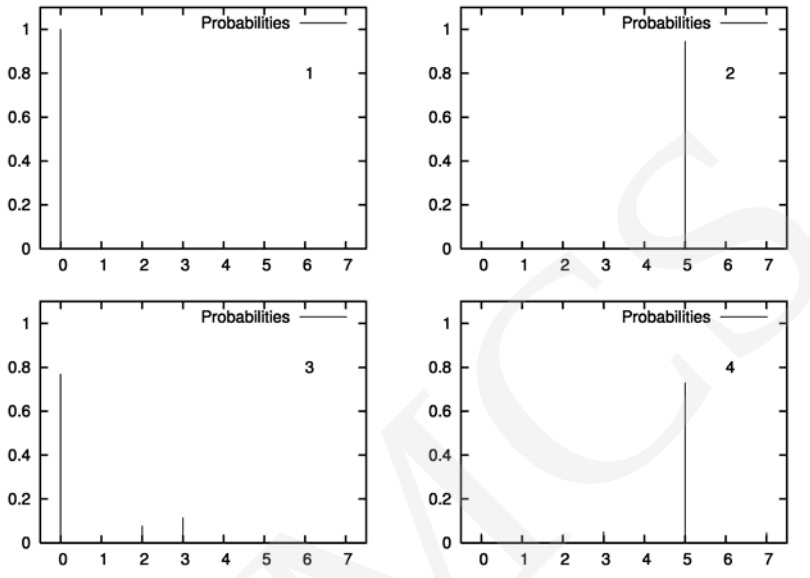


Fig. 3. Example of graphical output generated by quantum-octave

4. Teaching scenarios

"Introduction to quantum algorithms using the example of Deutch's algorithm"

This class introduces students to quantum algorithms and is divided into three parts: theoretical computations, preparation of quantum circuits, implementation and simulation of those circuits in QCSI.

Deutsch's algorithm [17] can be described as follows: let us assume one is in the possession of black box, called oracle, that computes the function $f:\{0,1\}\to\{0,1\}$ One does not know if function f is constant $f(0)=f(1)$, or balanced $f(0)=\overline{f(1)}$. In the classical case we have to apply the oracle twice to find out which kind of function f is. Deutsch's algorithm allows to reduce application of oracle to only one.

Graphically this algorithm can be represented as in Fig. 4.

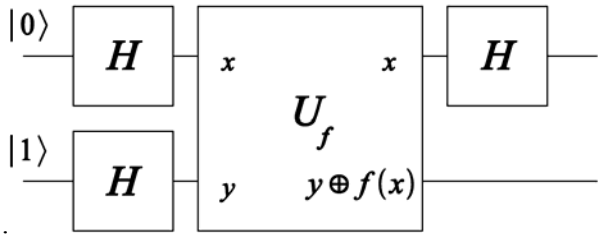


Fig. 4. Graphical representation of Deutsch algorithm

After measurement of the first qubit if one obtains outcome $|0\rangle$ it means that f is constant, if $|1\rangle$ it means that f is balanced. Students have to make calculations for all four possible functions and try to write Deutsch's algorithm symbolically. Afterwards students should prepare quantum circuit for one chosen function and implement it in QCSI.

Example of implementation for the identity function is shown in Fig. 5.

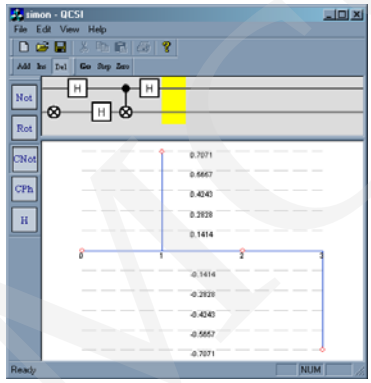


Fig. 5. Simulation of Deutsch algorithm in QCSI

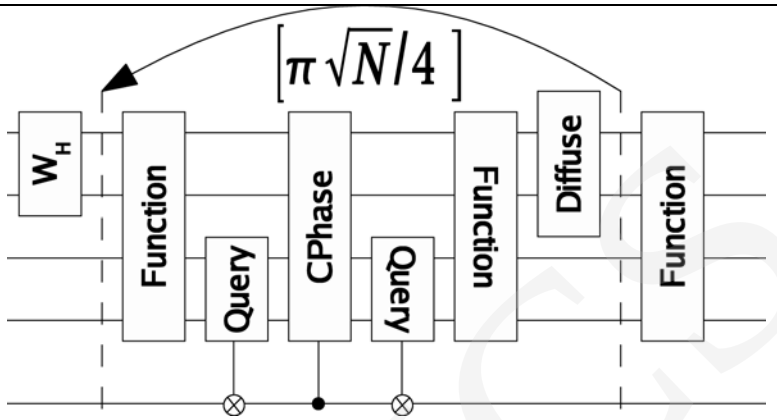
In the last part of this class students should observe the time evolution of quantum register gate by gate, compare simulation results with theoretical calculations and draw conclusions about discovered advantages and constraints of quantum computing.

"Analysis of Grover's algorithm and its modification to database search with use of QCL and visualization program QCL2eps"

Grover's algorithm [18] is historically the first major achievement of quantum informatics. During this class students get familiar with Quantum Computing Language and try to implement useful quantum program. Use of visualization program helps understand the process of translation from the high level code to the quantum circuit performed by QCL interpreter.

This class is composed of two parts: analysis of Grover's algorithm code, modification of this code for database search.

During the analysis of Grover's idea one can ask the question: "Why in Gorver's algorithm do we search the element whose index we already know?" To understand how this algorithm can be useful one has to make some modification of the oracle [19]. Graphical representation of this idea is presented in Fig. 5.



W_H – Welsh Hadamard gate,
Function – unitary representation of database,
Query – gate marking searching element,
CPhase – controlled phase shift gate,
Diffuse – diffusion operator.

Gates between dashed lines are repeated $\left\lfloor \frac{\pi}{4}\sqrt{N} \right\rfloor$ times, where N is length of the first quantum register.

Fig. 6. Graphical representation of Grover's algorithm

We make the assumption that students already know and understand Grover's algorithm from the lecture. The first part of this class is the analysis of code provided by QCL library. For easier understanding how it works students may use the visualization tool described earlier.

The second part consists of implementation of quantum operator representing database and of its use in the Grover's algorithm code. In this case database should be represented as a small truth table. We call its first column – keys and the second – values. It's important for keys and values to be unique. Otherwise, it would not be possible to implement that table as an unitary operator. Database can be coded with the use only of controlled-not and negation gates as reversible permutational circuit.

To test and verify functionality of created code students should do it using the simulator integrated with QCL. Concluding one should make some reflections about similarity of the problem solved above with the method of breaking symmetrical cryptographic system such as DES.

5. Future application of simulators in quantum informatics

Today simulation of quantum machines is the only possibility of testing of quantum algorithms. But even when real quantum computer will be constructed simulators will still be useful, because of the ability to observe quantum states

without performing its destructive measurement. This fact can be used for example to debug quantum programs.

Acknowledgements

The research was supported by Grant No. 7 T11C 017 21 from State Committee for Scientific Research in Poland.

References

[1] Benett C. H., Brassard G., *Quantum cryptography: public key distribution and coin tossing*, Proseedings of IEEE International Conference on Computers, Systems and Signals Processing, Bangalore, India, (1984) 175.
[2] Gawron P., Grochla K., *Kwantowa dystrybucja kluczy w sieciach optycznych*, Studia Informatica, 24(2b)(54) (2003) 223.
[3] http://www.idquantique.com/
[4] Deutsch D., *Quantum Theory, the Church-Turing Principle, and the Universal Quantum Computer*, Proceedings of the Royal Society of London, A400 (1985) 97.
[5] Landauer R., *Information is Physical*, Physics Today, 44 (1991) 23.
[6] Leung D.W., Chuang I.L., Yamaguchi F., Yamamoto Y., *Efficient implementation of coupled logic gates for quantum computation*, Physical Review A, 61, (2000).
[7] Kwiat P.G., Mitchell J.R., Schwindt P.D.D., White A.G., *Grover's search algorithm: an optical approach*, J. Mod. Optics, 47 (2000) 257.
[8] Oemer B., *Structured Quantum Programming*, http://tph.tuwien.ac.at/~oemer/qcl.html
[9] Oemer B., *A Procedural Formalism for Quantum Computing*, Master Thesis technical physics, TU Vienna, (1998).
[10] Oemer B., *Quantum Programming in QCL*, Master Thesis computing science, TU Vienna, (2000).
[11] Bettelli S., Serafini L., Calarco T., Toward an architecture for quantum programming, arXiv:cs.PL/0103009
[12] Gawron P., *Symulacja komputerów kwantowych*, http://www.iitis.gliwice.pl/zksi/publikacje.php.pl
[13] Quantum-octave homepage, http://quantum-octave.sf.net/
[14] Octave homepage http://www.octave.org/
[15] QuCalc homepage: http://crypto.cs.mcgill.ca/QuCalc/
[16] Libquantum homepage: http://www.enyo.de/libquantum/
[17] Nielsen M.A., Chuang I.L., *Quantum Computation and Quantum Information*, Cambridge University Press, (2002).
[18] Grover L., *A fast quantum mechanical algorithm for database search,* arXiv:quant-ph/9605043, (1996).
[19] Ross, D.A.: A Modification of Grover's Algorithm as a Fast Database Search, arXiv:quant-ph/9807078