

Kwantowe języki programowania

Robert Nowotniak

IV rok informatyki (Sztuczna Inteligencja)

WWW: <http://robert.nowotniak.com/>

e-mail: <*robercik@stud.ics.p.lodz.pl*>

Plan referatu

- Możliwość symulacji komputera kwantowego
- Obecne języki kwantowe
- Biblioteki kwantowe dla Perla
- Język QCL
 - Informacje ogólne
 - Podstawy składni
 - Proste przykłady
 - Dalsze informacje
- Podsumowanie

Symulowanie komputera kwantowego

- Czy można symulować takie komputery?
- W jaki sposób?
- Prace Alana Turinga (1936)
 - Wykonywanie **dowolnego procesu algorytmicznego** może być symulowane za pomocą odpowiedniego, abstrakcyjnego mechanizmu (maszyna Turinga, rachunek lambda Church'a).
 - Maszyna Turinga
 - Matematyczny model komputera
 - $MT = \langle Q, \Sigma, \delta, \Gamma, q_0, B, F \rangle$
- Kwantowa Maszyna Turinga (David Deutsch, 1985)

Symulowanie komputera kwantowego

- Maszyna Turinga pozwala symulować także algorytmy kwantowe, jednak ta symulacja ma wykładniczą złożoność obliczeniową.
- **Liniowy przyrost liczby kubitów w rejestrze kwantowym powoduje wykładniczy przyrost wymiaru przestrzeni stanów takiego rejestru.**
- Symulując komputer kwantowy, nie mamy dobrodziejstw „kwantowej równoległości”, którą musimy symulować z powyższą złożonością.

Języki programowania kwantowego

- Zrealizowane implementacje:
 - **QCL** (interpreter języka napisany w C++)
 - Biblioteki Perla:
 - **Quantum::Entanglement**
 - Umożliwia kwantowe splątanie zmiennych
 - jako przykład zaimplementowany jest algorytm Shora
 - **Quantum::Superposition**
 - libquantum (biblioteka dla języka C)
 - Q-gol (w języku CaML + interfejs Tcl/Tk)
- Niezaimplementowane (jedynie projekty):
 - qGCL, QML, Quantum C

Quantum::Entanglement

```
#!/usr/bin/perl

use Quantum::Entanglement qw(:DEFAULT :complex :QFT);

my $c = entangle(0.7071, 0, 0.7071*i, 1); # sqrt(2) / 2 * (|0> + i|1>)
my $d = entangle(0.7071, 0, 0.7071, 1); # sqrt(2) / 2 * (|0> + |1>)

#
# $e <- 0.5*|0*0> + 0.5*i|0*1> + 0.5*|1*0> + 0.5*i|1*1>
# Kwantowe splatanie zmiennej $e z $c oraz $d
#

$e = $c * $d;

#
# Pomiar wartości $e. Redukcja funkcji falowej
#
if ($e == 1) {
    # Rezultatem pomiaru jest, że $c == 1 , $d == 1
    print '$e == 1';
    print "\n";
}
else {
    # Pozostałe przypadki
    print '$e != 1';
    print "\n";
}
```

Quantum::Entanglement

- Funkcja `entangle()` zwraca wartość, która jest superpozycją wielu wartości z poszczególnymi amplitudami prawdopodobieństwa.
- Działania na kilku zmiennych będących w superpozycji stanów powodują kwantowe splątanie tych zmiennych.
- Przeciążone operatory konwersji i relacji dla zmiennej będącej w superpozycji powodują redukcje odpowiadającej jej funkcji falowej i dekoherencję stanu tej zmiennej **oraz** zmiennych z nią splątanych.
- Specjalne funkcje `p_op` i `p_func` służą do wykonywania operacji na zmiennych będących w superpozycji (odpowiedniki bramek kwantowych).

libquantum API

```
quantum_reg quantum_new_qureg(MAX_UNSIGNED initval, int width);  
void quantum_print_qureg(quantum_reg reg);
```

```
void quantum_cnot(int control, int target, quantum_reg *reg);  
void quantum_toffoli(int controll1, int control2, int target, quantum_reg *reg);
```

```
void quantum_sigma_x(int target, quantum_reg *reg);  
void quantum_sigma_y(int target, quantum_reg *reg);  
void quantum_sigma_z(int target, quantum_reg *reg);  
void quantum_r_x(int target, float gamma, quantum_reg *reg);  
void quantum_r_y(int target, float gamma, quantum_reg *reg);  
void quantum_r_z(int target, float gamma, quantum_reg *reg);
```

```
void quantum_hadamard(int target, quantum_reg *reg);
```

```
MAX_UNSIGNED quantum_measure(quantum_reg reg);  
int quantum_bmeasure(int pos, quantum_reg *reg);
```

libquantum

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <quantum.h>

int main ()
{
    quantum_reg reg;
    int wynik;

    srand(time(0));

    reg = quantum_new_quireg(0, 1);
    quantum_hadamard(0, &reg);

    wynik = quantum_bmeasure(0, &reg);

    printf("Odczytana wartość: %d!\n", wynik);

    return 0;
}
```

Quantum Computation Language

- Autor: PhD Bernhard Ömer
 - <http://tph.tuwien.ac.at/~oemer/qcl.html>
- Aktualna wersja: qcl 0.6.3
 - Ostatnie wydanie: **14 grudnia 2006**
 - (aktywnie rozwijany!)
- Wersje binarne dostępne obecnie tylko dla systemu GNU/Linux na architekturze:
 - i686
 - **AMD64 (układy EM64T)**
- Wolne oprogramowanie (licencja GNU/GPL)

Język QCL

- Język QCL:
 - Strukturalny
 - Imperatywny
 - Kwantowy
 - Język trzeciej generacji
- Składnia przypominająca język C
- Podobne podstawowe typy danych
- Typ danych **qureg** – rejestr kwantowy
- Wbudowane podstawowe operacje kwantowe
 - Bramka Hadamarda (H(), Mix())
 - Bramka NOT, CNOT
 - Bramka fazy (CPhase)
 - Definiowanie własnych operatorów kwantowych (**qfunc**)

Język QCL

Uruchomienie interpretera QCL:

```
$ qcl --bits=5
QCL Quantum Computation Language (5 qubits, seed 1166577494)
[0/5] 1 |0>
```

„Podejrzanie” zawartości pamięci kwantowej:

```
qcl> dump
: STATE: 0 / 5 qubits allocated, 5 / 5 qubits free
1 |0>
qcl> _
```

Utworzenie nowego 2-kubitowego rejestru kwantowego:

```
qcl> qureg a[2];
qcl> dump a;
: SPECTRUM a: <0,1>
1 |0>
qcl> _
```

Negacja stanu rejestru kwantowego:

```
qcl> Not(a);
[2/5] 1 |3>
qcl> dump
: STATE: 2 / 5 qubits allocated, 3 / 5 qubits free
1 |3>
qcl> _
```

Język QCL

Utworzenie superpozycji stanów czystych tego rejestru
(bramka Hadamarda):

```
qcl> H(a) ;  
[2/5] 0.5 |0> - 0.5 |1> - 0.5 |2> + 0.5 |3>  
qcl> _
```

Pomiar wartości rejestru - dekoherencja stanu.

```
qcl> measure a ;  
[2/5] -1 |1>  
qcl> _
```

Ponowne „podejrzenie” wartości rejestru:

```
qcl> dump a ;  
: SPECTRUM a: <0,1>  
1 |1>  
qcl>
```

Przykład: Algorytm Deutsch'a w QCL

- Algorytm służy do zbadania funkcji:
- $F : \{0, 1\} \rightarrow \{0, 1\}$
- Istnieją dokładnie cztery takie funkcje:
- dwie funkcje różnowartościowe i dwie stałe

	F ₁	F ₂	F ₃	F ₄
0	1	0	0	1
1	0	1	0	1

Sprawdzenie wartości funkcji dla jednego argumentu nie pozwala rozstrzygnąć, z jaką funkcją mamy do czynienia

Ale algorytm kwantowy – może to wykonać.

Przykład

- Definicje wszystkich czterech funkcji w QCL:

```
// ~x
qfunct F1(qureg out, quconst in) {
    CNot(out, in);
    Not(out);
}
```

```
// x
qfunct F2(qureg out, quconst in) {
    CNot(out, in);
}
```

```
// 0
qfunct F3(qureg out, quconst in) {
}
```

```
// 1
qfunct F4(qureg out, quconst in) {
    Not(out);
}
```

Algorytm Deutsch'a w QCL

```
print "Algorytm Deutsch'a";

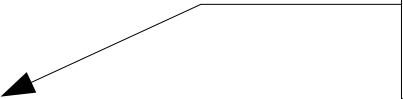
qureg in[1];
qureg out[1];
int result;

reset;
Not(out);
H(out);
H(in);
F4(out, in);
H(in);
H(out);

measure in, result;

if (result == 0) {
    print "Funkcja stala";
}
else {
    print "Funkcja roznowartosciowa";
}
```

Jednokrotne wywołanie
jednej ze zdefiniowanych
funkcji.



Język QCL

- Podstawowe instrukcje:
 - reset
 - include
 - print
 - input
 - qureg
 - qfunc
 - measure
 - dump

Literatura

- 1) <http://www.quantiki.org/> , <http://ocw.mit.edu/OcwWeb/>
- 2) Prace Bernharda Ömera , <http://tph.tuwien.ac.at/~oemer/qcl.html>
 - 1) ***A Procedural Formalism for Quantum Computing***
 - 2) ***Quantum Programming in QCL***
 - 3) ***Structured Quantum Programming***
- 3) Artykuł w jednym z numerów *Software Developer Journal*
- 4) R.Penrose – Nowy umysł cesarza
- 5) S.Imre, F.Balazs - Quantum Computing and Communications: Engineering Approach
- 6) Samuel Braunstein - Quantum Computing (Wiley Press, 1999)
- 7) **Mika Hirvensalo - Algorytmy Kwantowe, WSiP 2004**
- 8) G.Chen, R.Brylinski - Mathematics of Quantum Computation
- 9) D.C.Marinescu - Lectures Notes on Quantum Computing and Quantum Information Theory for Non-Physicists

Literatura

- 10) Michel Le Bellac - Quantum Physics, Cambridge Press
- 11) R.Shankar - Principles of Quantum Mechanics, 1994
- 12) J.J.Sakurai - Modern Quantum Mechanics
- 13) Asher Peres - Quantum Theory: Concepts and Methods
- 14) Devid J.Griffiths - Introduction to Quantum Mechanics, 1995
- 15) Andre Berthiaume - Quantum Computation
- 16) John Preskill - Quantum Computing: Pro and Con, 1997
- 17) S.K.Shukla, R.I.Bahar - Nano, Quantum and Molecular Computing
- 18) M.A.Nielsen, I.L.Chuang - Quantum Computation and Quantum Information (Cambridge University Press, 2000)
- 19) Jozef Gruska - Quantum Computing (McGraw-Hill, 1999)
- 20) C.Williams, S.Clearwater - Explorations in Quantum Computing

Dziękuję