Quantum Computation Language

Krzysztof Piecuch, 332534

April 13, 2015

Poniższy tekst (i przykładowy fragment kodu w QCL) jest zaczerpnięty ze strony języka, tj. http://tph.tuwien.ac.at/~oemer/qcl.html

QCL to wysokopoziomowy język programowania dla komputerów kwantowych, który może być wykonywany na maszynie o dowolnej architekturze. Składnia przypomina składnię popularnych języków proceduralnych (Pascal, C). Ideą jest emulacja kwantowych obliczeń na komputerach o dowolnej architekturze. Jako, że kwantowe obliczenia to dość złożone zgadanienie, to poniższy tekst sporo upraszcza i nie podaje (kompletnych) powodów. Całość jest opisana na w/w stronie, tekst jest tylko przekładem i streszczeniem "najważniejszych" pod kątem implementacji faktów z dostępnych tam dokumentów. Nie zamieszczam definicji pojęć i stosowanej notacji, ponieważ jest standardowa dla dziedziny nauki i zamieszczona tutaj, a dokument ma być tylko pobieżnym opisem najważniejszych cech języka. Czasami są użyte nieścisłe, niestandardowe pojęcia, jednak w takim przypadku należy posłużyć się intuicją.

O co chodzi z kwantowymi obliczeniami?

Każdy klasyczny program może zostać zamodelowany jako drzewo decyzyjne, gdzie węzeł odpowiada binarnemu stanowi i ma jedno lub więcej dzieci-stanów. Deterministyczna maszyna Turinga pozwala na tylko jedno przejście między stanami, więc ścieżka obliczeń jest z góry określona. Na probabilistycznej maszynie Turinga przejścia są określone przez prawdopodobieństwa i jedno z możliwych przejść jest wybierane losowo, jednakże dopóki tego nie sprawdzimy, maszyna Turinga znajduje się we wszystkich stanach na tym samym poziomie drzewa jednocześnie (superpozycja). Większość kwantowych algorytmów korzysta z tego posługując się superpozycją kubitów, a po przeszukaniu drzewa sprawdza się wartości kubitów, dzięki czemu (przy pewnych założeniach i ograniczeniach) jesteśmy w stanie stwierdzić, jaka byłaby ścieżka klasycznego algortmu. Operacja sprawdzenia wartości kubitu jest łączna i przemienna, kolejność, w jakich sprawdzamy wartości jest bez znaczenia. Oprócz tego, kwantowe obliczenia są potężniejsze od probabilistycznych dzięki interferencji destruktywnej, która zwiększa prawdopodobieństwo "dobrych" stanów i obniża prawdopodobieństwo tych, które są "mało obiecujące".

Stan maszyny i programu Stan programu Pamięć kwantowego komputera to zazwyczaj kombinacja 2-stanowych podsystemów zwanych bitami kwantowymi (kubitami). Zawartość pamięci to stan wszystki kubitów. W tym przypadku posługujemy się określeniem "stan maszyny", zamiast "stan programu", który to definiował "sytuację" w klasycznych językach programowania.

Stan maszyny n kubitowego komputera kwantowego to wektor w przestrzeni Hilberta $H = C^{2^n}$, jednakże, w związku z destruktywnym działaniem pomiaru (qubit może przyjmować obydwa stany jednocześnie, dopóki się nie sprawdzi jego stanu) nie można bezpośrednio sprawdzić jego wartości i nie jest dostępny z poziomu QCL. Dostępne są jednak funkcje *load*, *save*, *dump*, które służą do wykonywania operacji bez ingerencji w stan maszyny. QCL oddziela klasyczny stan programu i kwantowy stan programu, można w nim tworzyć w pełni klasyczne programy.

Kwantowe rejestry QCL używa rejestrów kwantowych jako interfejsu, z którego może korzystać użytkownik, żeby zmienić stan maszyny. Kwantowy rejestr dla programisty to "wskaźnik" na ciąg parami różnych kubitów, więc jest traktowany jako zwykła zmienna. Wszystkie unitarne operacje, które operują na stanie maszyny oraz sprawdzenie stanu przyjmowały kwantowe rejestry (oprócz *reset*) jako argumenty. Jako, że n-kubitowy komputer kwantowy pozwala na $\frac{n!}{(n-m)!}$ różnych m kubitowych rejestrów, jakakolwiek operacja lub pomiar może skutkować $\frac{n!}{(n-m)!}$ różnym operacjom na stanie maszyny. To wymusza, żeby każda unitarna operacja była wykonalna na pojedynczych kubitach i wymaga fizycznej architektury do pomiaru pojedynczych kubitów.

Kwantowa sterta Pozwala na alokację i dealokację kubitowych rejestrów. Rozmieszczenie kubitów w rejestrach jest niewidoczne dla użytkownika, dzięki czemu określenie pozycji kubitów nie musi być określane przez użytkownika. Jako, że stan maszyny można opisać stanem wspólny mdwóch niezależnych maszyn - jednej, która ma całą zaalokowaną pamieć a druga całą niezaalokowaną, to definiowane algorytmy kwantowe nie zależą od całkowitej liczby kubitów.

Alokacja rejestrów Rejestry są alokowane na zasadzie stosu - kubity są wkładane na stos w czasie alokacji i zdejmowane (pamięć jest zwalniana) gdy opuszczamy "zasięg zmiennej".

Tymczasowe rejestry kwantowe Aby mieć pewność, że nie uszkodzi się struktury kwantowej sterty, należy zapewnić wyczyszczenie rejestru przed jego dealokacją. Funkcje kwantowe pozwalają na deklaracje lokalnych wartości jako tymczasowych, dzięki czemu poprawne zwalnianie pamięci jest przezroczyste dla programisty i jest zapewniane przez specjalny algorytm w interpreterze.

Kwantowe wyrażenia

wyrażenie	opis	rejestr
a	referencja	$\langle a_0, a_1, \cdots, a_n \rangle$
$a[i]$	kubit	$\langle a_i \rangle$
$a[i:j]$	podsłowo	$\langle a_i, \cdots, a_j \rangle$
$a[i\backslash j]$	podsłowo	$\langle a_i, \cdots, a_{i+j-1} \rangle$
$a\&b$	połączenie	$\langle a_0, \cdots, a_n, b_0, \cdots, b_m \rangle$

Kwantowe funkcje (subroutines) Oprócz podstawowych *procedur* i $funkcji$ QCL ma funkcje dla bezstanowych operatorów unitarnych (*operator*) i pseudo-klasycznych operatorów ($qufunct$). QCL pozwala na odwracanie operatorów i musi zarządzać pamięcią tymczasową w funkcjach kwantowych, więc określono następujące ograniczenia na efekty uboczne działające na stan programu i kwantowy stan maszyny:

typ funkcji	stan programu	stan maszyny	rekursja
procedura	wszystkie	wszystkie	tak
operator	żadne	unitarne	nie
$qufunct$	żadne	pseudo-klasyczne	nie
$funkcje$	żadne	żadne	tak

Z powyższej tabelki wynika także, że funkcja nie może wołać tylko funkcje tego samego lub niższego rzędu. Matematyczna semantyka *operatorów* i $funkcji$ wymaga, żeby wszystkie wywołania były bezstanowe. Stąd stan programu nie może być zmieniany przez te funkcje oraz ich wywołanie nie może zależeć od globalnego stanu programu, który zawiera globalne zmienne, opcje i stan wewnętrznego generatora liczb losowych.

Zewnętrzna edycja stanu maszyny Język nie może posiadać wbudowanych metod do zmiany stanu maszyny, ponieważ wymagałoby to założeń dotyczących architektury symulowanego komputera kwantowego. Stąd QCL udostępnia *operator* oraz $qufunct$, które poprzedzone słowem *extern* pozwalają na wykonywanie instrukcji poza językiem.