QWIRE: A QRAM-Inspired Quantum Circuit Language

Jennifer Paykin, Robert Rand, Steve Zdancewic
University of Pennsylvania

Classical Computer

QWIRE
- small
- linear types
- strongly normalizing
- higher-order functions
- dependent types
- analysis of circuits

Quantum Computer
- initialize qubits
- apply unitary gates
- measure qubits

Quantum circuits
results of measurement

Host Language

QWIRE
C ::= output p
| p ← new t; C
| x ← measure p; C
| x ← C; C'
| unbox t p
p ::= () | w | (p₁,p₂)

W ::= 1
| qubit
| W₁ ⊗ W₂

Quantum Teleportation

teleport :: CIRC(qubit,qubit)
teleport = box q =>
(a,b) ← unbox bell00 ();
(q,a) ← unbox alice (q,a);
(x,y) ← measure (q,a);
unbox (bob x y) b

bell00 :: CIRC(1,qubit ⊗ qubit)
bell00 = box () =>
(a,b) ← new (0,0);
gate H on a;
gate CNOT on (a,b);
output (a,b)

alice :: CIRC(qubit@qubit@qubit)
alice = box (q,a) =>
gate CNOT on (q,a);
gate H on q;
output (q,a)

bob :: Bit -> Bit -> CIRC(qubit,qubit)
bob x y = box b =>
b ← unbox (if y then X_CIRC else ID) b
b ← unbox (if x then Z_CIRC else ID) b
output b

Meta-Programming with Circuits

inSequence :: CIRC(W₁,W₂) -> CIRC(W₂,W₃)
inSequence c₁ c₂ = box w₁ =>
w₂ ← unbox c₁ w₁;
w₃ ← unbox c₂ w₂;
output w₃

inParallel :: CIRC(W₁,W₂) -> CIRC(W₁',W₂')
inParallel c c' = box (w₁,w₁') =>
w₂ ← unbox c w₁;
w₂' ← unbox c' w₁';
output (w₂,w₂')

fourier :: forall (n :: Nat+). CIRC(Qubits n, Qubits n)
fourier 1 = box w => gate H on w; output w
fourier (n+1) = box (q,w) =>
w ← unbox fourier n w;
unbox rotations (n+1) n' (q,w)

Host Language Dependent Types

fourier 4

Normalization
unbox (box p ⇒ C) p' → C{p'/p}
p ← output p; C → C{p'/p}
p ← (gate u on p; C'); C →
gate u on p; y ← C';

fourier 2

Classical Computer

Quantum Computer

quantum
results of measurement

Host Language

Bell 00

H

Qubits

Wires

unbox

box

Quantum Teleportation

teleport

alice

bob

bell00

H

inSequence

inParallel

fourier

Host Language Dependent Types

fourier 1

fourier 2

fourier 3

fourier 4

rotations 4, 3