Recent developments in quantum programming

Jarosław Miszczak (IITiS PAN)
with contributions from Ryan LaRose (Michigan State University)

COST Meeting IC1405 'Reversible Computation'
Malta, March 11-14, 2019

Outline

Goals

Quantum programming

Gate-level programming platforms

Utilization of reversibility

Summary/What next?

Goals

- Review the recent progress in quantum software
- Demonstrate the utilization of reversibility
- Compare different approaches

Quantum programming

Quantum programming

What is quantum programming?

Quantum programming is a process that leads from an original formulation of a computing problem to **executable** quantum computer programs.

Quantum programming

Why bother?

- Utilize quantum computers → gate level quantum programming platforms
- Create a new programming language with non-classical elements... → high-level quantum programming
- ...or a language for describing quantum mechanics. → categorical quantum mechanics

Quantum programming

How to do quantum programming?

By the level of abstraction

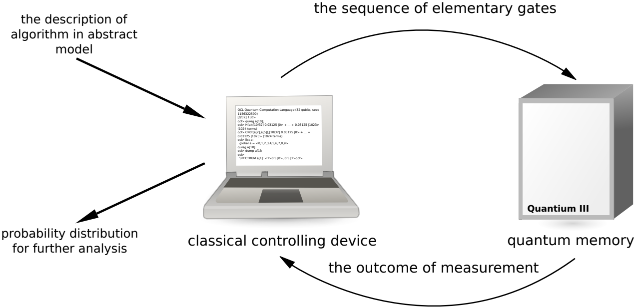
- Level 0: Manipulation of matrices.
- Level 1: Gate-level programming.
- Level 2: High-level programming.

In practice the existing systems are mixtures of those approaches, but the main trend is gate-level programming.

Quantum programming

Concept of QRAM

QRAM ≡ Quantum Random Access Machine



Quantum programming

Advantages of QRAM

What is this good for?

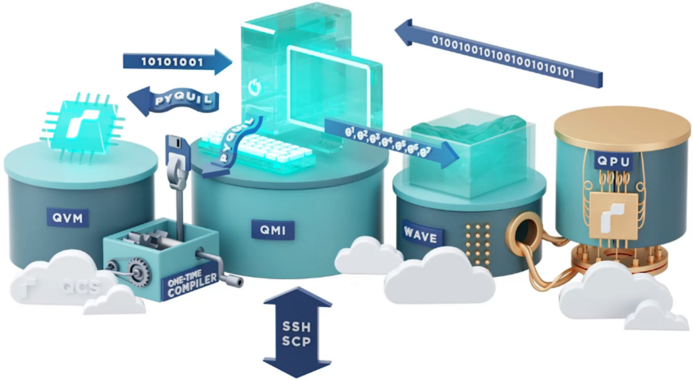
- data abstraction ≡ allocation of quantum memory
- compound quantum operations ≡ functions encapsulating sequence of quantum gates or quantum primitives
- quantum control of quantum operations ≡ generalized CNOT
- classical control of quantum operations ≡ loops, ifs etc. mixed with quantum code

(E. Knill. Conventions for quantum pseudocode. Technical report, Los Alamos National Laboratory, 1996. Technical Report)

Gate-level programming platforms

Gate-level programming platforms

QRAM in the cloud (a.k.a QRAM 2.0)

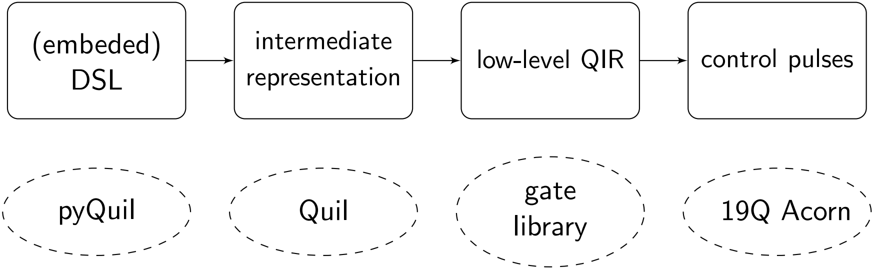


Quantum Cloud Services

Source: http://rigetti.com/qcs

Gate-level programming platforms

Software architecture



(See more at https://pyquil.readthedocs.io/)

Gate-level programming platforms

Basic example in pyQuil

```
from pyquil import Program, get_qc
from pyquil.gates import H, CNOT, MEASURE

qvm = get_qc('2q-qvm') # connect to QVM with 2 quibts

prg = Program() # build the program
out = prg.declare('ro', 'BIT', 2) # declare classical memory

# construct the code
prg += H(0)
prg += CNOT(0, 1)
prg += MEASURE(0, out[0])

# construct the intermediate representation
exe = qvm.compile(prg)

# run the code
res = qvm.run(exe)
```

Gate-level programming platforms

Quantum middleware

Quantum computers are expensive → utilize a layer of intermediary software.

Usually this is

- embedded domain specific language → Python with library of functions
- data abstraction → allocation of classical and quantum registers based on qu(b|d)its
- classical control of quantum memory → by using host language
- quantum functions → custom gates defined by matrices or compound statements

Gate-level programming platforms

Quantum middleware has its advantages...

- easy to learn and use
- auto-magic quantum memory management and circuit generation
- integration with classical machine
- library of gates can build and re-used

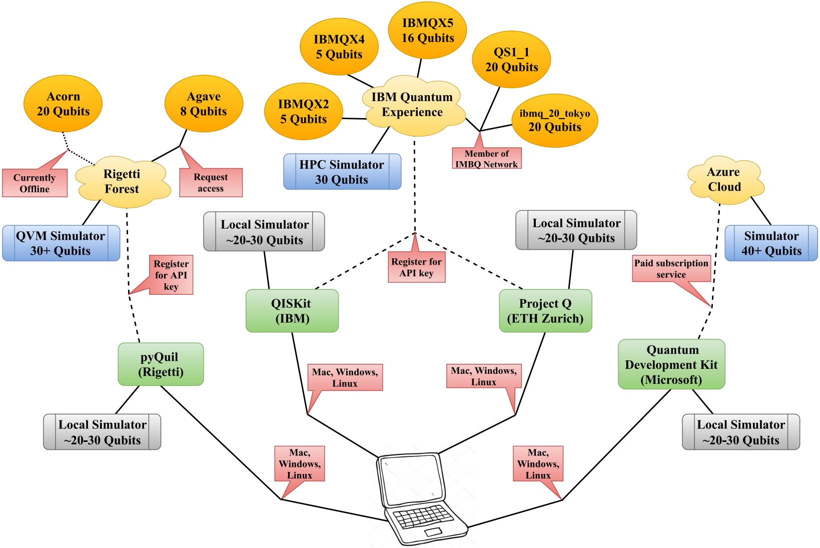
Gate-level programming platforms

...and its disadvantages

- very similar to code using matrices (Level 0)
- lack of expressivity

Gate-level programming platforms

What are the options?



Gate-level programming platforms

What are the options?

In terms of software → hardware

- QISKit → IBM
- pyQuil → Rigetti
- ProjectQ → IBM (and other backends)
- Quantum Development Kit → Microsoft (???)
- Cirq → Google (???)
- QX Simulator → Intel/QuTech (???)

(D-Wave System does not count here, it is based on a different model.)

1QBit | BOHR | CQC Cambridge Quantum Computing | ENTROPICA LABS
GTN LTD | HQS QUANTUM SIMULATIONS | OTI | ProteinQure
QC WARE | QULAB | Q^x Branch | rahko
RIVERLANE | SolidStateAI | STRANGE WORKS | ZAPATA

(Source: https://www.microsoft.com/en-us/quantum/quantum-network)

Warning

Quantum computing is an exciting field that has caught the imagination of the public. This is a good thing. But if the quantum computing effort starts to mingle fact with fiction, then the entire effort loses its credibility.
– Umesh Vaziriani (April 7th, 2007)

https://www.scottaaronson.com/blog/?p=225

Utilization of reversibility

Utilization of reversibility

From circuit-level to high-level

- ProjetQ – circuit-level
- IQu – mixed-model

Utilization of reversibility

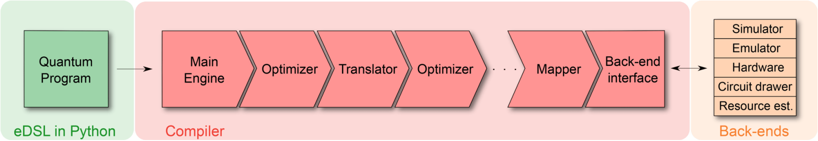
Example 1: ProjectQ

- Python library developed by ETH (`https://projectq.ch/`)
- offers various targets
  - hardware (IBM Q Experience)
  - C++ simulator
  - graphical circuit representation
  - resource counter (???)

Utilization of reversibility

Example 1: ProjectQ

- also based on the concept of quantum middleware
- more flexible and not tied to particular vendor



Utilization of reversibility

Example 1: ProjectQ

Nice features

- Natural (for physicist) syntax for executing quantum gates.
- Meta instructions for quantum-controlled quantum operations and support for reversible execution

Utilization of reversibility

Example 1: ProjectQ – Basic usage

```
from projectq import MainEngine
from projectq.ops import H, Measure
from projectq.backends import IBMBackend
import projectq.setups.ibm

eng = MainEngine(IBMBackend(),
      engine_list=projectq.setups.ibm.get_engine_list())

q2 = eng.allocate_qubit()

# quantum instructions
H | q2
Measure | q2

eng.flush() # this requires login and password
print(int(q2))
```

Utilization of reversibility

Example 1: ProjectQ – Quantum-controlled quantum gates

Meta instruction `Control`

- quantum controlled quantum gates

```
with Control(eng, qc):
    H | qt
```

- can be used within control blocks

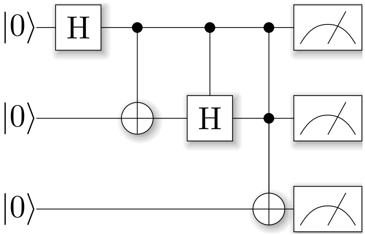
```
with Control(eng, qc):
    X | qt
    H | qt
    with Control(eng, qt):
        X | qtt
```

Utilization of reversibility

Example 1: ProjectQ – Quantum-controlled quantum gates

Meta instruction `Control`

Execution of the code is based on the state of quantum register.



Utilization of reversibility

Example 1: ProjectQ – Reversibility

Meta instruction Dagger

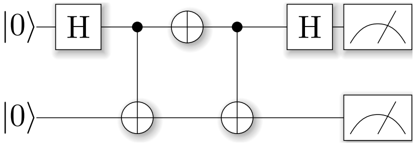
- reverse execution of the quantum code

```
def compute_block(q):
    H | q[0]
    CNOT | (q[0],q[1])

compute_block(qr)

X | qr[0]

with Dagger(eng):
    compute_block(qr)

All(Measure) | qr
```

Utilization of reversibility

Example 1: ProjectQ – Reversibility

Meta instruction `Dagger`



Utilization of reversibility

Example 1: ProjectQ – Reversibility

Meta instruction Compute/Uncompute

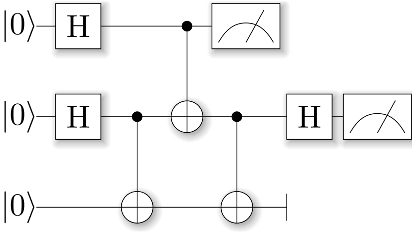
- reverse execution with ancilla management

```
with Compute(eng):
    q3 = eng.allocate_qubit()
    H | q1
    CNOT | (q1, q3)

CNOT | (q0, q1)

Uncompute(eng)
```

Utilization of reversibility

Example 1: ProjectQ – Reversibility

Meta instruction `Compute/Uncompute`



Utilization of reversibility

Domain specific languages

High-level programming

Domain specific language with data and function abstraction.

- IQu (https://quilabverona.wordpress.com/)
- QCL (http://tph.tuwien.ac.at/~oemer/qcl.html)
- LanQ (http://lanq.sourceforge.net/)
- QPL anc cQPL (https://arxiv.org/abs/quant-ph/0511145)
- Scaffold (https://github.com/epiqc/ScaffCC)

Utilization of reversibility

Example 2: IQu – programming quantum circuits

- Only specification available, developed by Verona group (https://quilabverona.wordpress.com/).
- Prototypical language that combines quantum commands and states with higher order features
- Part of the language focused on circuit management. Quantum memory is not considered.

Utilization of reversibility

Example 2: IQu – programming quantum circuits

Quantum co-processor is as a black-box that receiving a suitable circuit, gives back a total measurement executed on the final state.

- In IQu quantum circuits are treated as classical data.
- They can be composed sequentially ($\circ_\circ$) or in parallel ($\|$)
- Circuit expressions can utilize `iter` and `reverse`.

Utilization of reversibility

Example 2: IQu – programming quantum circuits

Let us assume that we have `H`, `CNOT` and `X` gates available.

```
iter H 2 H
H ‖ H ‖ H # same as above
CNOT ‖ ID # CNOT on first two qubits
iter ID 2 X # same as ID ‖ X ‖ X
reverse CNOT ‖ ID
```

Summary/What next?

- interest in quantum computing exploded, but the application of quantum algorithms is still unclear
- computing platforms shape the syntax of the programming languages
- we have more focus on the manipulation of the circuits (construction of quantum circuits/executable code)
- this leads to support for reversible being added — syntax (IQu), methods (pyQuil), or language extensions (ProjectQ)

Summary/What next?

- J. Miszczak, Quantum programming tutorial: slides and code examples, https://github.com/jmiszczak/qprog-tutorial
- R. LaRose, *Overview and Comparison of Gate Level Quantum Software Platforms*, arXiv:1807.02500
  - NISQAI – library developed for applications on near-term quantum computers. https://github.com/quantumai-lib/nisqai
  - More at: https://www.ryanlarose.com/
- D. Koch, L. Wessing, P.M. Alsing, Introduction to Coding Quantum Algorithms: A Tutorial Series Using Qiskit, arXiv:1903.04359
- Quantum programming schools planned by Verona group and QuSoft@Riga (http://qusoft.lu.lv/).

Thank you.