Quantum Programming Languages and Frameworks

A Survey by Magne Tenstad

Selected Topics of Software Technology: Quantum Computing (VU 716.204)

June 27, 2024

Outline

- Features and Challenges
- A Quantum Hello World
- Top Frameworks
- Other Noteworthy Languages
- On the Need of a Quantum-Oriented Paradigm
- Final Remarks

Features and Challenges

Features

- Qubits and quantum gates
- Superposition and entanglement
- Measurement
- (Integration with classical code)

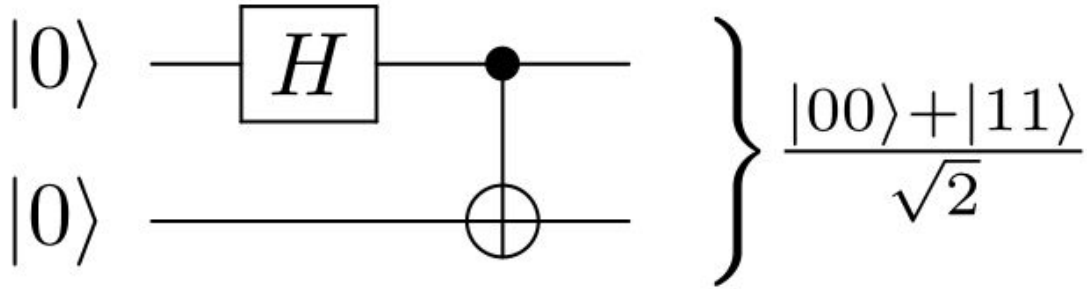
Challenges

- Design useful abstractions
- Quantum error handling
- Debugging
- Hardware support

The Bell State, a Quantum Hello World

- Two qubits
- One hadamard gate
- One controlled-not gate

- Illustrates entanglement and superposition
- Measurement: either 00 or 11



Source

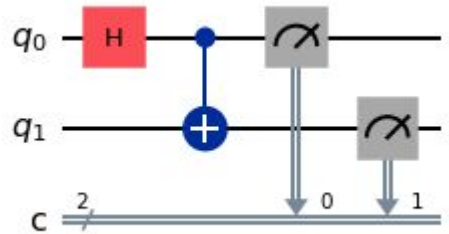
Top Frameworks

- **Qiskit** IBM, 2017
- **Q#** Microsoft, 2017
- **Pyquil** Rigetti, 2017
- **Cirq** Google, 2018

These were excellently covered by Hannah Jud in this course in 2022.

Qiskit (IBM, 2017)

```python
from qiskit import QuantumCircuit, transpile
from qiskit_aer import AerSimulator

qc = QuantumCircuit(2, 2)
qc.h(0)
qc.cx(0, 1)
qc.measure([0, 1], [0, 1])

backend = AerSimulator()
qc_compiled = transpile(qc, backend)
print(backend.run(qc_compiled, shots=10)
    .result().get_counts(qc_compiled))
```



Q# (Microsoft, 2017)

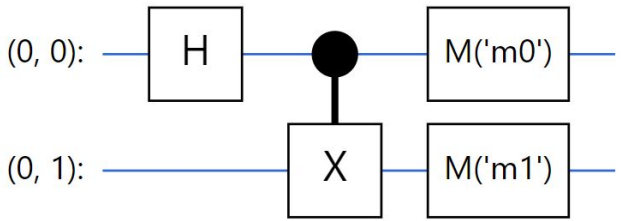
```
open Microsoft.Quantum.Intrinsic;
open Microsoft.Quantum.Canon;
open Microsoft.Quantum.Measurement;

namespace BellState
{
    @EntryPoint()
    operation BellStateWithMeasurement() : (Result, Result) {
        using (qubits = Qubit[2]) {
            H(qubits[0]);
            CNOT(qubits[0], qubits[1]);
            let m0 = M(qubits[0]);
            let m1 = M(qubits[1]);
            return (m0, m1);
        }
    }
}
```

Pyquil (Rigetti, 2017)

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

qvm = get_qc("2q-qvm")

p = Program()
p += H(0)
p += CNOT(0, 1)
ro = p.declare("ro", "BIT", 2)
p += MEASURE(0, ro[0])
p += MEASURE(1, ro[1])
p.wrap_in_numshots_loop(10)

print(qvm.run(p).get_register_map()["ro"].tolist())
```

Cirq (Google, 2018)

```
from cirq import CX, Circuit, GridQubit, H,
Simulator, measure

q0 = GridQubit(row=0, col=0)
q1 = GridQubit(row=0, col=1)
c = Circuit()
c.append(H(q0))
c.append(CX(q0, q1))
c.append(measure(q0, key="m0"))
c.append(measure(q1, key="m1"))

print(Simulator().run(c, repetitions=10))
```



Observations on the Top Frameworks

- From 2017-2018
- Most are based on Python
- Enable quantum circuit construction
- Associated with services from large companies
    - IBM Qiskit Runtime, Microsoft Azure, Rigetti Quantum Cloud Services, Google Quantum Computing Service

➔ Unlike classical programming languages, it seems that *services* fuel development and popularity of quantum programming languages.

Other Noteworthy Languages

- **QCL** 1998
- **Quipper** 2013
- **OpenQASM** 2017
- **Silq** 2020
- **QHDL** 2023
- **Rhyme** 2024

QCL (1988)

- Recognized as first high-level quantum programming language

```
procedure quRoulette() {
  qureg q[5];
  int field;
  int number;
  input "Enter field number:",field;
  repeat {
    Mix(q);
    measure q,number;
    reset;
  } until number<=36;
  if field==number {
    print "Number",number,"You won!";
  } else {
    print "Number",number,"You lose.";
  }
}
```

Table 2.12: **roulette.qcl** quantum roulette

Quipper (2013)

- “Scalable, expressive, functional and higher-order”
- Based on Haskell

```
import Quipper

bellState :: Circ (Bit, Bit)
bellState = do
  q0 <- qinit False
  q1 <- qinit False

  hadamard_at q0
  qnot_at q1 `controlled` q0

  m0 <- measure q0
  m1 <- measure q1

  return (m0, m1)

main :: IO ()
main = print_simple Preview bellState
```

OpenQASM (2017)

- Represents universal physical circuits
- Compilation target for other languages
- Elements of C and assembly languages

```
OPENQASM 2.0;
include "qelib1.inc";
qreg q[2];
creg c[2];
h q[0];
cx q[0], q[1];
measure q[0] -> c[0];
measure q[1] -> c[1];
```

Silq (2020)

- High-level language
- Intuitive semantics
- Supports safe, automatic uncomputation

```
d := a || b || c;
```
Silq

```
with_computed (OR a b) $
  \t -> OR t c
```
Quipper

```
using(t=Qubit()){
  OR(a,b,t);
  OR(t,c,d);
  Adjoint OR(a,b,t);
}
```
Q#

Figure 1. Benefit of Silq's automatic uncomputation.

```
cTri := 0:int[rrbar];
for j in [0..rrbar) {
  for k in [j+1..rrbar) {
    if ee[tau[j]][tau[k]]
       && eew[j] && eew[k]{
      cTri += 1;
} } }
```
Silq

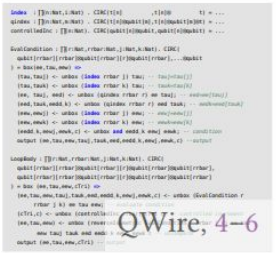
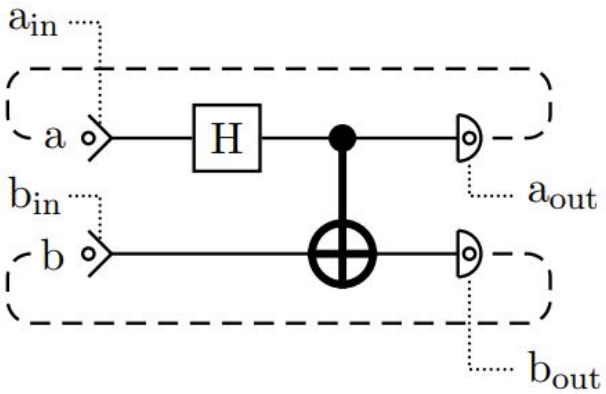


Figure 2. Comparing Silq to Quipper and QWire code, more readable version in App. A.

QHDL (2023)

- Low-level circuit description language for QC
- Inspired by VHDL




```
1  -- A pair of qbits that can be prepared in a Bell state
2
3  library qhdl;
4
5  use qhdl.std.all;
6
7  entity bellstate is
8    port (
9      clk: in bit;
10     a_in, b_in: in bit;
11     a_out, b_out: out bit
12     );
13 end entity bellstate;
14
15 architecture quantum of bellstate is
16   signal reg_a, reg_b, had_a, not_a, not_b,
17       meas_a, meas_b: qbit;
18 begin
19   setter_a: qset port map ( clk => clk, d => meas_a,
20       q => reg_a, set => a_in );
21   setter_b: qset port map ( clk => clk, d => meas_b,
22       q => reg_b, set => b_in );
23   hadamat_a: qhadamard port map ( d => reg_a, q=> had_a );
24   entangle: qcnot port map ( c_in => had_a, c_out => not_a,
25       d => reg_b, q => not_b );
26   measure_a: qmeasure port map ( clk => clk, d => not_a,
27       q => meas_a, result => a_out );
28   measure_b: qmeasure port map ( clk => clk, d => not_b,
29       q => meas_b, result => b_out );
30 end architecture quantum; -- bellstate
```

Figure 5. Bellstate circuit in QHDL.

Rhyme (2024)

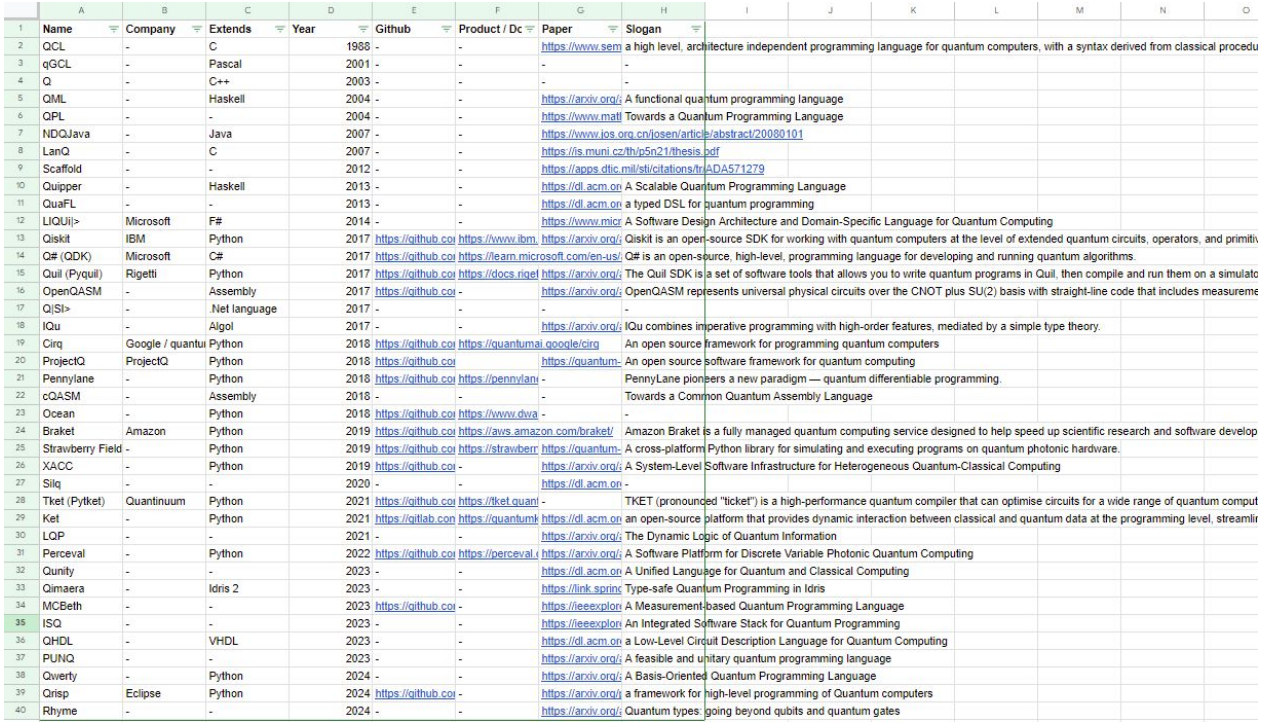
- Quantum types as extensions of classical bits, integers, floats, characters, arrays, and strings.
- || operator creates superposition of classical values

```
qfloat f = 2.5 || 3.5;
qfloat g = 3.14159 || 2.71828;
qref r = &f || &g;

ref x = r; // r collapses, x is either &f or &g
float y = *x; // *x is a qfloat, it collapses, y
↪  holds outcome

print(y);
```

On the Need of a Quantum-Oriented Paradigm (QOP)

- Paper from 2023 by Shaukat Ali and Tao Yue
- “Specialized backgrounds are required to build QC applications, limiting the maximum exploitation of QC’s potential.”
- QOP shall enable cost-effective and intuitive development, independent of low-level quantum mechanics (e.g., superposition and entanglement).
- Proposes *encapsulation*, *abstractions*, and *separation of concerns* as features of the QOP.

List of Quantum Languages and Frameworks



Name	Company	Extends	Year	Github	Product / Dc	Paper	Slogan
QCL	-	C	1988	-	-	https://www.sem	a high level, architecture independent programming language for quantum computers, with a syntax derived from classical procedu
qGCL	-	Pascal	2001	-	-	-	-
Q	-	C++	2003	-	-	-	-
QML	-	Haskell	2004	-	-	https://arxiv.org/	A functional quantum programming language
QPL	-	-	2004	-	-	https://www.matl	Towards a Quantum Programming Language
NDQJava	-	Java	2007	-	-	https://www.jos.org.cn/josen/article/abstract/20080101	
LanQ	-	C	2007	-	-	https://is.muni.cz/th/p5n21/thesis.pdf	
Scaffold	-	-	2012	-	-	https://apps.dtic.mil/sti/citations/tr/ADA571279	
Quipper	-	Haskell	2013	-	-	https://dl.acm.or	A Scalable Quantum Programming Language
QuaFL	-	-	2013	-	-	https://dl.acm.or	a typed DSL for quantum programming
LIQUi|>	Microsoft	F#	2014	-	-	https://www.micr	A Software Design Architecture and Domain-Specific Language for Quantum Computing
Qiskit	IBM	Python	2017	https://github.cor	https://www.ibm.	https://arxiv.org/	Qiskit is an open-source SDK for working with quantum computers at the level of extended quantum circuits, operators, and primiti
Q# (QDK)	Microsoft	C#	2017	https://github.cor	https://learn.microsoft.com/en-us/		Q# is an open-source, high-level, programming language for developing and running quantum algorithms.
Quil (Pyquil)	Rigetti	Python	2017	https://github.cor	https://docs.rigel	https://arxiv.org/	The Quil SDK is a set of software tools that allows you to write quantum programs in Quil, then compile and run them on a simulato
OpenQASM	-	Assembly	2017	https://github.cor	-	https://arxiv.org/	OpenQASM represents universal physical circuits over the CNOT plus SU(2) basis with straight-line code that includes measureme
Q|SI>	-	.Net language	2017	-	-	-	-
IQu	-	Algol	2017	-	-	https://arxiv.org/	IQu combines imperative programming with high-order features, mediated by a simple type theory.
Cirq	Google / quantu	Python	2018	https://github.cor	https://quantumai	google/cirg	An open source framework for programming quantum computers
ProjectQ	ProjectQ	Python	2018	https://github.cor		https://quantum-	An open source software framework for quantum computing
Pennylane	-	Python	2018	https://github.cor	https://pennylan	-	PennyLane pioneers a new paradigm — quantum differentiable programming.
cQASM	-	Assembly	2018	-	-	-	Towards a Common Quantum Assembly Language
Ocean	-	Python	2018	https://github.cor	https://www.dwa	-	-
Braket	Amazon	Python	2019	https://github.cor	https://aws.amazon.com/braket/		Amazon Braket is a fully managed quantum computing service designed to help speed up scientific research and software develop
Strawberry Field	-	Python	2019	https://github.cor	https://strawberr	https://quantum-	A cross-platform Python library for simulating and executing programs on quantum photonic hardware.
XACC	-	Python	2019	https://github.cor	-	https://arxiv.org/	A System-Level Software Infrastructure for Heterogeneous Quantum-Classical Computing
Silq	-	-	2020	-	-	https://dl.acm.or	-
Tket (Pytket)	Quantinuum	Python	2021	https://github.cor	https://tket.quant	-	TKET (pronounced "ticket") is a high-performance quantum compiler that can optimise circuits for a wide range of quantum comput
Ket	-	Python	2021	https://gitlab.con	https://quantumk	https://dl.acm.or	an open-source platform that provides dynamic interaction between classical and quantum data at the programming level, streamlir
LQP	-	-	2021	-	-	https://arxiv.org/	The Dynamic Logic of Quantum Information
Perceval	-	Python	2022	https://github.cor	https://perceval.(	https://arxiv.org/	A Software Platform for Discrete Variable Photonic Quantum Computing
Qunity	-	-	2023	-	-	https://dl.acm.or	A Unified Language for Quantum and Classical Computing
Qimaera	-	Idris 2	2023	-	-	https://link.spring	Type-safe Quantum Programming in Idris
MCBeth	-	-	2023	https://github.cor	-	https://ieeexplor	A Measurement-based Quantum Programming Language
ISQ	-	-	2023	-	-	https://ieeexplor	An Integrated Software Stack for Quantum Programming
QHDL	-	VHDL	2023	-	-	https://dl.acm.or	a Low-Level Circuit Description Language for Quantum Computing
PUNQ	-	-	2023	-	-	https://arxiv.org/	A feasible and unitary quantum programming language
Qwerty	-	Python	2024	-	-	https://arxiv.org/	A Basis-Oriented Quantum Programming Language
Qrisp	Eclipse	Python	2024	https://github.cor	-	https://arxiv.org/	a framework for high-level programming of Quantum computers
Rhyme	-	-	2024	-	-	https://arxiv.org/	Quantum types: going beyond qubits and quantum gates

https://docs.google.com/spreadsheets/d/17GZWMUrBYf23iieLb7x7utYwPUTWOcY7z2bHBhqnWpU

Final Remarks

- Availability of hardware steers development
- Quantum languages' requirements differ from classical languages
  - Still, popular quantum frameworks are implemented in classical languages
- Most are circuit-based, but Silq, Rhyme and QOP and bringing new ideas
- Heavily researched topic, new approaches are proposed each year

References

- P. Singh *et al*., "A Survey on Available Tools and Technologies Enabling Quantum Computing," in *IEEE Access*, vol. 12, pp. 57974-57991, 2024, doi: https://doi.org/10.1109/ACCESS.2024.3388005
- Shaukat Ali and Tao Yue. 2023. On the Need of Quantum-Oriented Paradigm. In Proceedings of the 2nd International Workshop on Quantum Programming for Software Engineering (QP4SE 2023). Association for Computing Machinery, New York, NY, USA, 17–20. https://doi.org/10.1145/3617570.3617868
- Language-specific references listed in this spreadsheet.
- Implementations of the bell state in Q#, Quipper and OpenQASM were done by OpenAI's ChatGPT.