Counting Qubits and Gates: Resource Analysis in Quantum Programming Languages

Ugo Dal Lago

ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

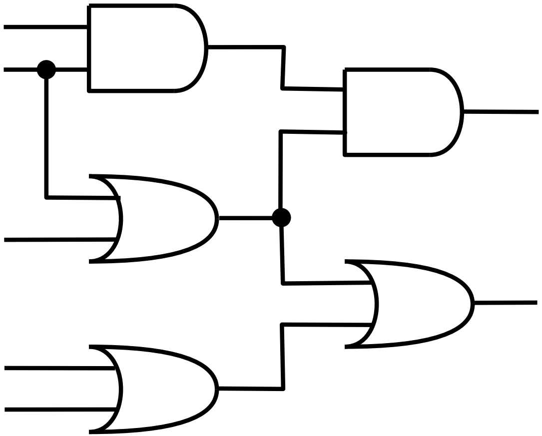
Inria informatiques mathématiques

IFIP WG 2.2 Meeting, September 26h 2025

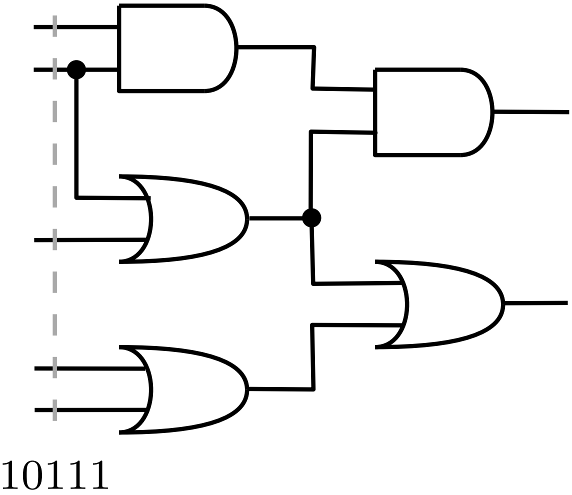
Part I

Quantum Computing

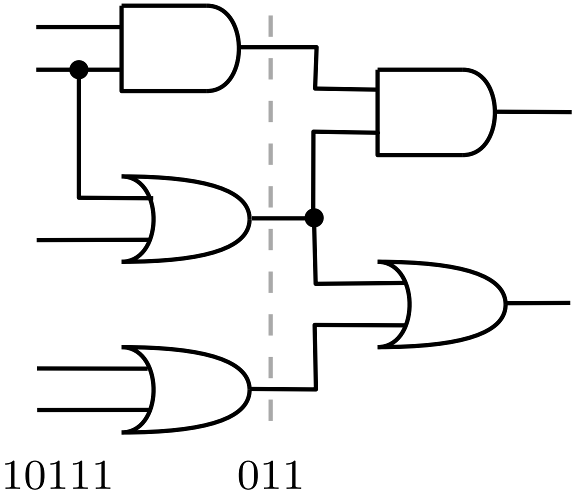
Boolean Circuits



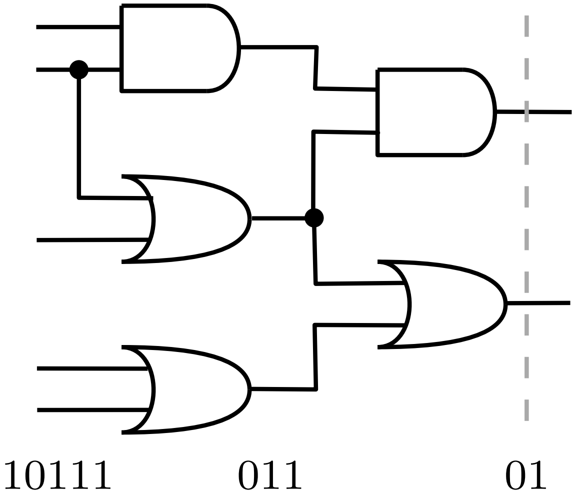
Boolean Circuits



Boolean Circuits

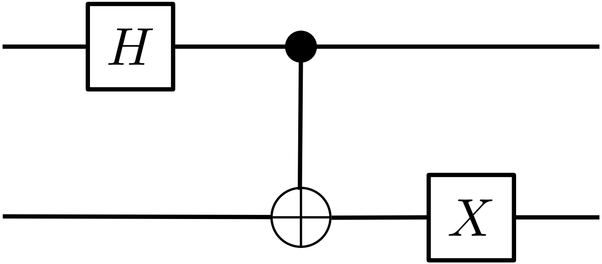


Boolean Circuits

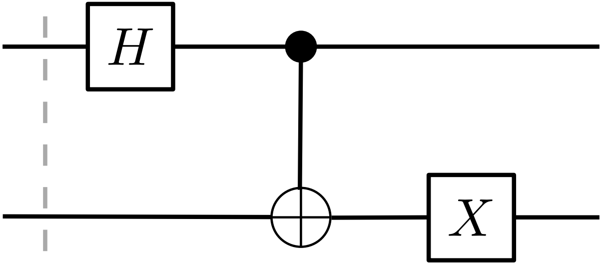


Quantum Circuits

Quantum Circuits

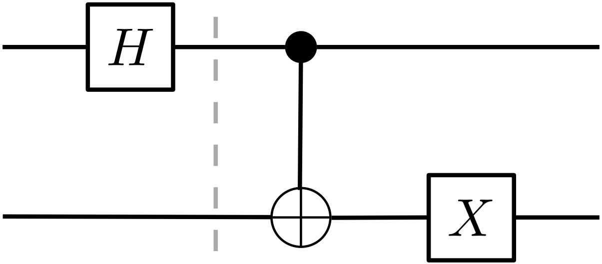


Quantum Circuits



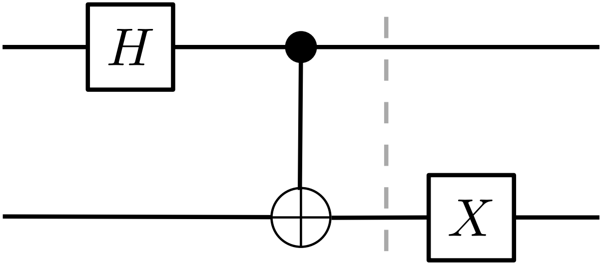
$$(1,0) \otimes (1,0) = (1,0,0,0)$$

Quantum Circuits



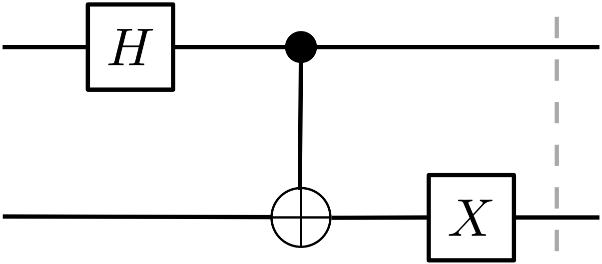
$$\left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right) \otimes (1, 0) = \left(\frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}}, 0\right)$$

Quantum Circuits



$$\left(\frac{1}{\sqrt{2}}, 0, 0, \frac{1}{\sqrt{2}} \right)$$

Quantum Circuits



$$\left(0,\ \frac{1}{\sqrt{2}},\ \frac{1}{\sqrt{2}},\ 0\right)$$

Three Non-Intuitive Phenomena

Superposition

Three Non-Intuitive Phenomena

- There is a form of **parallelism** which has nothing to do with uncertainty.
- The number of coefficients is **exponential** in the number of wires.

Superposition

Three Non-Intuitive Phenomena

Superposition

Entanglement

Three Non-Intuitive Phenomena

States like

$$\left(0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0\right)$$

cannot be decomposed through the tensor product.

Superposition

Entanglement

Three Non-Intuitive Phenomena

Superposition

Entanglement

Negative Interference

Three Non-Intuitive Phenomena

Superposition

Entanglement

Negative Interference

- Coefficients can be **negative**.
- There is a phenomenon of cancellation which is not there in probabilistic computing.

Quantum Computing is Nice

Shor (1994)

Cryptography

HHL (2008)

Machine Learning

Finance

Optimization Problems

Grover (1996)

Unstructured Search

Chemistry

VQE (2014)

Quantum Computing is Hard

Quantum Computing is Hard

- Current quantum architectures can handle only a **very limited** number of qubits.
- We are still in the so-called **NISQ** era.
- **Quantum advantage** is not there yet, despite the breakthrough theoretical advances.

Quantum Computing is Hard

News in focus

algorithms that harness these indicators to estimate a person's 'biological age', which can be higher or lower than their chronological age[3].

Another hallmark of ageing is a shift in the proteins that the body produces. To explore how organs age, Oh and his colleagues first analysed nearly 5,000 proteins in blood samples from 1,398 healthy adults. They identified about 850 proteins that originated mainly from a single organ and trained a machine-learning algorithm to predict a person's age on the basis of the levels of these proteins. They validated their model using blood samples from more than 4,000 other people.

The results showed that an organ's biological age is linked to disease risk. For example, roughly 2% of participants had accelerated heart ageing – that is, their levels of blood proteins relating to heart ageing differed substantially from those of other people of the same age. Having a prematurely old heart was linked to a 250% increased risk of heart failure, the authors found.

Marking time

Researchers have used epigenetic markers to show that the pace of organ ageing varies between individuals[4]. But the link between epigenetic changes and ageing is unclear, says Matt Kaeberlein, a specialist in the biology of ageing and chief executive of Optispan, a biotechnology company in Seattle, Washington. Proteins are "much closer to the downstream

IBM RELEASES FIRST-EVER 1,000-QUBIT QUANTUM CHIP

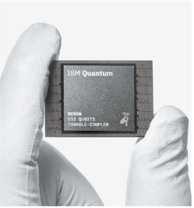
The company will now focus on developing smaller, more reliable processors.

By Davide Castelvecchi

IBM has unveiled the first quantum computer with more than 1,000 qubits – the equivalent of the digital bits in an ordinary computer. But the company says that it will now shift gears and focus on making its machines more error-resistant rather than larger.

For years, IBM has been following a quantum-computing road map that roughly doubled the number of qubits every year. The chip unveiled on 4 December, called Condor, has 1,121 superconducting qubits arranged in a honeycomb pattern. It follows on from IBM's other record-setting, bird-named machines, including a 127-qubit chip called Eagle, released in 2021 and a 433-qubit one called Osprey, announced last year.

Quantum computers promise to perform certain computations that are beyond the reach of classical computers. They will do so



RYAN LAVINE FOR IBM

IBM's Heron quantum processor.

2023). The company says that it will now focus on building chips designed to hold a few qLDPC-corrected qubits in just 400 or so

QUBITS ARE SCARCE

Quantum Computing is Hard

How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits

Craig Gidney[1] and Martin Ekerå[2]

[1]Google Inc., Santa Barbara, California 93117, USA
[2]KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden
Swedish NCSA, Swedish Armed Forces, SE-107 85 Stockholm, Sweden

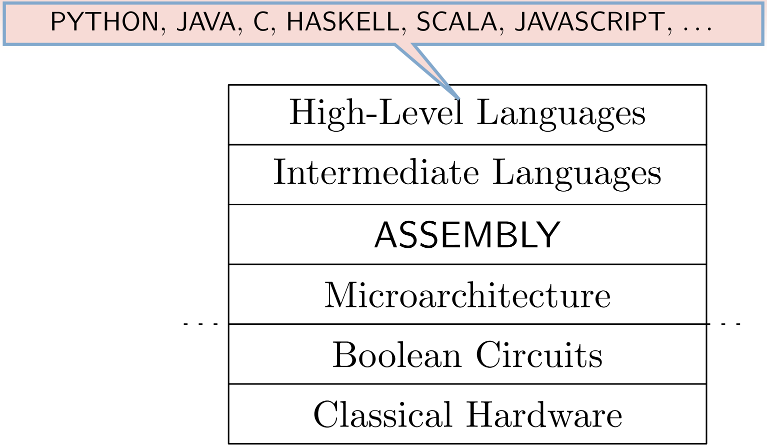
We significantly reduce the cost of factoring integers and computing discrete logarithms in finite fields on a quantum computer by combining techniques from Shor 1994, Griffiths-Niu 1996, Zalka 2006, Fowler 2012, Ekerå-Håstad 2017, Ekerå 2017, Ekerå 2018, Gidney-Fowler 2019, Gidney 2019. We estimate the approximate cost of our construction using plausible physical assumptions for large-scale superconducting

A DREAM

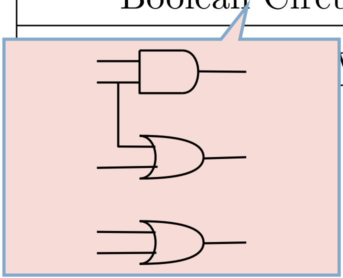
Part II

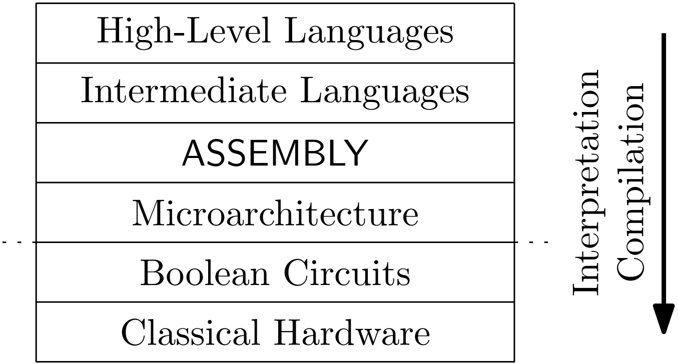
Quantum Programming Languages

High-Level Languages
Intermediate Languages
ASSEMBLY
Microarchitecture
Boolean Circuits
Classical Hardware

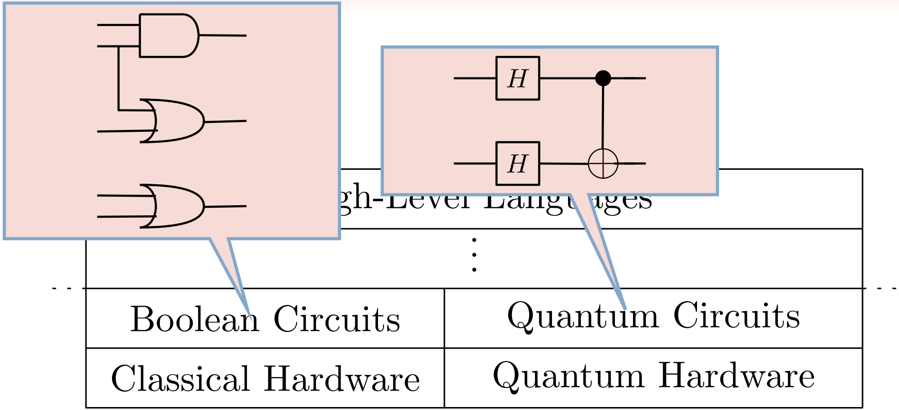

PYTHON, JAVA, C, HASKELL, SCALA, JAVASCRIPT, ...
High-Level Languages
Intermediate Languages
ASSEMBLY
Microarchitecture
Boolean Circuits
Classical Hardware

High-Level Languages
Intermediate Languages
ASSEMBLY
Microarchitecture
Boolean Circuits
ware




High-Level Languages
Intermediate Languages
ASSEMBLY
Microarchitecture
Boolean Circuits
Classical Hardware
Interpretation
Compilation

High-Level Languages	
⋮	
Boolean Circuits	Quantum Circuits
Classical Hardware	Quantum Hardware


H
H
Boolean Circuits
Quantum Circuits
Classical Hardware
Quantum Hardware

High-Level Languages

⋮

ircuits

rdware

- How could we *construct* quantum high-level programs?
- How could we compile an high-level program down to a *mixed* architecture?
- How to take advantage of the presence of quantum circuits, and of the computation power they provide?

THE QRAM MODEL

Conventions for Quantum Pseudocode

LANL report LAUR-96-2724

E. Knill

knill@lanl.gov, Mail Stop B265
Los Alamos National Laboratory
Los Alamos, NM 87545

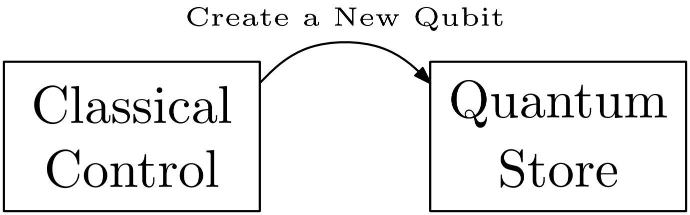
June 1996

Abstract

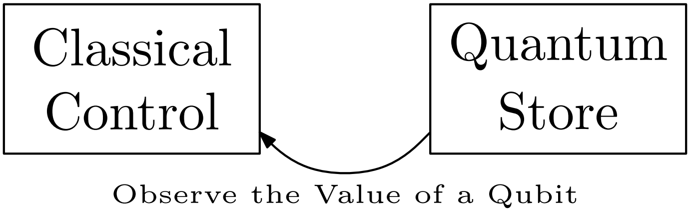
A few conventions for thinking about and writing quantum pseudocode are proposed. The conventions can be used for presenting any quantum algorithm down to the lowest level and are consistent with a quantum random access machine (QRAM) model for quantum computing. In principle a formal version of quantum pseudocode could be used in a future extension of a conventional language.
Note: This report is preliminary. Please let me know of any suggestions, omissions or errors so that I can correct them before distributing this work more widely.

Quantum Data and Classical Control

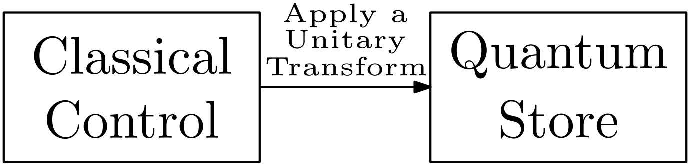
Quantum Data and Classical Control



Quantum Data and Classical Control



Quantum Data and Classical Control



A Brief Survey of Quantum Programming Languages

Peter Selinger

Department of Mathematics, University of Ottawa
Ottawa, Ontario, Canada K1N 6N5
selinger@mathstat.uottawa.ca

Abstract. This article is a brief and subjective survey of quantum programming language research.

1 Quantum Computation

Quantum computing is a relatively young subject. It has its beginnings in 1982, when Paul Benioff and Richard Feynman independently pointed out that a quantum mechanical system can be used to perform computations [11, p.12]. Feynman's interest in quantum computation was motivated by the fact that it is computationally very expensive to simulate quantum physical systems on classical computers. This is due to the fact that such simulation involves the manipulation is extremely large matrices (whose dimension is exponential in the size of the quantum system being simulated). Feynman conceived of quantum computers as a means of simulating nature much more efficiently.

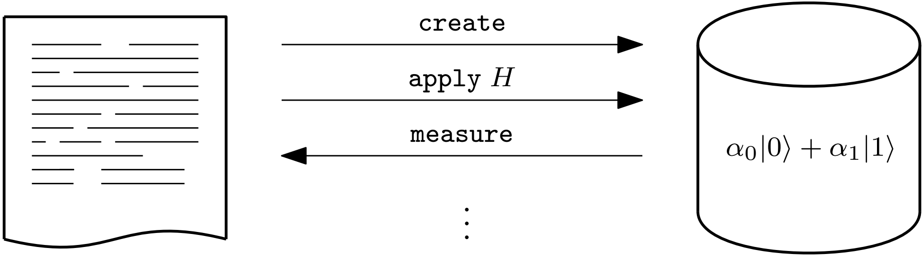
The evidence to this day is that quantum computers can indeed perform

A SURVEY

A Multitude of Idioms

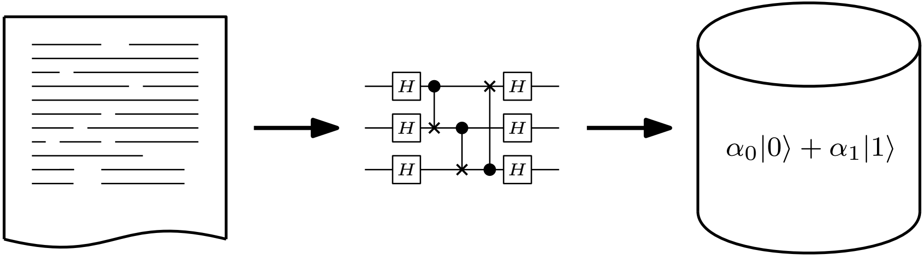
- **Functional Languages**
- **Imperative Languages**
- **Logic Programming Languages**
- **Quantum Constraint Programming Languages**
- ...

From QRAM. . .

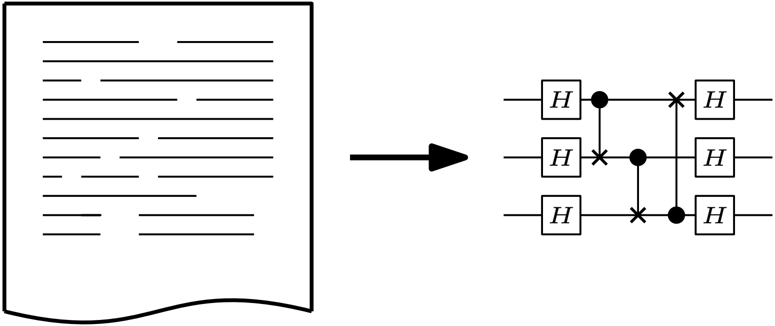


create
apply H
measure
$\vdots$
$\alpha_0|0\rangle + \alpha_1|1\rangle$

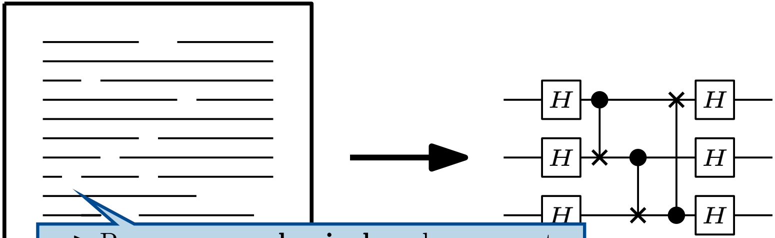
... to Reality



Quantum Circuit Description Languages



Quantum Circuit Description Languages



- Programs are **classical**, and are meant to **build** a quantum circuit, which is then executed by a quantum architecture, or simulated.
- Examples are `Qiskit`, `Cirq`, or `Quipper`.
- This is by far the **most popular** way to program quantum architectures.

Part III

Quantum Program Verification

QPV: Why and How

Rewards

- Testing quantum programs can be very **expensive**!
- Certain properties are arguably of **paramount importance**, like resource consumption.

QPV: Why and How

Rewards

- Testing quantum programs can be very **expensive**!
- Certain properties are arguably of **paramount importance**, like resource consumption.

Challenges

- The underlying computational model is simply **different**, and arguably more complicated.
- Besides the usual exponential blowup (there are $\Theta(2^n)$ states of size n), there is another one coming from **superposition**.
- Most quantum algorithms are only **approximately** correct, and many of them are meant to be run on **noisy** hardware.

QPV: Some Techniques

An Applied Quantum Hoare Logic

Li Zhou
Department of Computer Science
and Technology
Tsinghua University
China
zhou31416@gmail.com

Nengkun Yu
CQSI, FEIT
University of Technology Sydney
Australia
nengkunyu@gmail.com

Mingsheng Ying
CQSI, FEIT, University of Technology
Sydney, Australia
Institute of Software, CAS, China
Tsinghua University, China
Mingsheng.Ying@uts.edu.au

Abstract

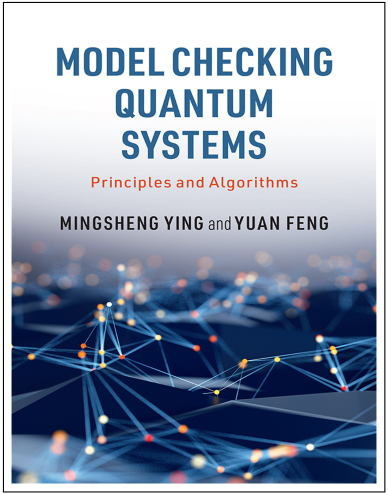
We derive a variant of quantum Hoare logic (QHL), called applied quantum Hoare logic (aQHL for short), by: (1) restricting QHL to a special class of preconditions and postconditions, namely projections, which can significantly simplify verification of quantum programs and are much more convenient when used in debugging and testing; and (2) adding several rules for reasoning about robustness of quantum programs, i.e. error bounds of outputs. The effectiveness of aQHL is shown by its applications to verify two sophisticated quantum algorithms: HHL (Harrow-Hassidim-Lloyd)

stimulated by rapid progress in the implementation of quantum computing hardware. Now people start to consider how to warrant correctness of quantum programs: debugging, testing or verification?

Quantum Hoare Logic: Indeed, attempts of developing Hoare-like logic for verification of quantum programs have been made in a series of papers [2, 6, 7, 9, 10, 14, 21, 29]. In particular, D'Hondt and Panangaden [13] proposed the notion of quantum weakest precondition, and Ying [37] established quantum Hoare logic (QHL for short) for both partial correctness and total correctness with (relative) completeness. More

PROGRAM LOGICS

QPV: Some Techniques



MODEL CHECKING

QPV: Some Techniques

Sized Types for Low-Level Quantum Metaprogramming

Matthew Amy

University of Waterloo, Waterloo, Canada
meamy@uwaterloo.ca

Abstract. One of the most fundamental aspects of quantum circuit design is the concept of *families* of circuits parametrized by an instance size. As in classical programming, metaprogramming allows the programmer to write entire families of circuits simultaneously, an ability which is of particular importance in the context of quantum computing as algorithms frequently use arithmetic over non-standard word lengths. In this work, we introduce metaQASM, a typed extension of the openQASM language supporting the metaprogramming of circuit families. Our language and type system, built around a lightweight implementation of *sized types*, supports subtyping over register sizes and is moreover type-safe. In particular, we prove that our system is strongly normalizing, and

TYPE SYSTEMS

Part IV

Quantum Resource Analysis

Part IV

Quantum Resource Analysis

- Deriving bounds on the **amount of resources** a quantum program needs.
- Given the **limitations** of current quantum hardware, this makes a lot of sense.
- This is an active research area, still in its infancy.

Implicit Computational Complexity

Implicit Computational Complexity

- This refers to machine-free characterizations of complexity classes which are based on logic or programming languages.
- In classical computing, this gave rise to characterizations of **many** complexity classes, like **P**, **NP**, **PSPACE**, etc.

Implicit Computational Complexity

Theoretical Computer Science 411 (2010) 377–409

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Quantum implicit computational complexity☆

Ugo Dal Lago[a,*], Andrea Masini[b], Margherita Zorzi[b]

[a] Dipartimento di Scienze dell'Informazione, Università di Bologna, Italy
[b] Dipartimento di Informatica, Università di Verona, Italy

ARTICLE INFO

Article history:
Received 7 October 2008
Received in revised form 5 June 2009
Accepted 15 July 2009
Communicated by P.-L. Curien

Keywords:
Quantum computation
Implicit computational complexity
Lambda calculus

ABSTRACT

We introduce a quantum lambda calculus inspired by Lafont's Soft Linear Logic and capturing the polynomial quantum complexity classes EQP, BQP and ZQP. The calculus is based on the "classical control and quantum data" paradigm. This is the first example of a formal system capturing quantum complexity classes in the spirit of implicit computational complexity — it is machine-free and no explicit bound (e.g., polynomials) appears in its syntax.

λ-CALCULUS

Implicit Computational Complexity

THE JOURNAL OF SYMBOLIC LOGIC
Volume 85, Number 4, December 2020

A SCHEMATIC DEFINITION OF QUANTUM POLYNOMIAL TIME COMPUTABILITY

TOMOYUKI YAMAKAMI

Abstract. In the past four decades, the notion of quantum polynomial-time computability has been mathematically modeled by quantum Turing machines as well as quantum circuits. This paper seeks the third model, which is a quantum analogue of the schematic (inductive or constructive) definition of (primitive) recursive functions. For quantum functions mapping finite-dimensional Hilbert spaces to themselves, we present such a schematic definition, composed of a small set of initial quantum functions and a few construction rules that dictate how to build a new quantum function from the existing ones. We prove that our schematic definition precisely characterizes all functions that can be computable with high success probabilities on well-formed quantum Turing machines in polynomial time, or equivalently uniform families of polynomial-size quantum circuits. Our new, schematic definition is quite simple and intuitive and, more importantly, it avoids the cumbersome introduction of the well-formedness condition imposed on a quantum Turing machine model as well as of the uniformity condition necessary for a quantum circuit model. Our new approach can further open a door to the descriptional complexity of quantum functions, to

FUNCTION ALGEBRAS

Implicit Computational Complexity

A Programming Language Characterizing Quantum Polynomial Time

Emmanuel Hainry, Romain Péchoux, and Mário Silva(✉)

Université de Lorraine, CNRS, Inria, LORIA, 54000 Nancy, France
{hainry,pechoux,mmachado}@loria.fr

Abstract. We introduce a first-order quantum programming language, named FOQ, whose terminating programs are reversible. We restrict FOQ to a strict and tractable subset, named PFOQ, of terminating programs with bounded width, that provides a first programming language-based characterization of the quantum complexity class FBQP. We finally present a tractable semantics-preserving algorithm compiling a PFOQ program to a quantum circuit of size polynomial in the number of input qubits.

QUANTUM CONTROL

Expectation Transformers

Expectation Transformers

- A well-studied generalization of Dijkstra's **predicate transformers**.
- It enables **quantitative reasoning** about the expectated value of random variables (e.g., the cost) of randomized programs.

Expectation Transformers

Quantum Expectation Transformers for Cost Analysis

Martin Avanzini
Inria Sophia Antipolis - Méditerranée
France
martin.avanzini@inria.fr

Georg Moser
Department of Computer Science
Universität Innsbruck
Innsbruck, Austria
georg.moser@uibk.ac.at

Romain Péchoux
Université de Lorraine, CNRS, Inria,
LORIA, F-54000 Nancy
France
romain.pechoux@inria.fr

Simon Perdrix
Université de Lorraine, CNRS, Inria,
LORIA, F-54000 Nancy
France
simon.perdrix@loria.fr

Vladimir Zamdzhiev
Université de Lorraine, CNRS, Inria,
LORIA, F-54000 Nancy
France
vladimir.zamdzhiev@inria.fr

ABSTRACT

We introduce a new kind of expectation transformer for a mixed *classical-quantum programming language*. Our semantic approach relies on a new notion of a cost structure, which we introduce and which can be seen as a specialisation of the Kegelspitzen of Keimel and Plotkin. We show that our weakest precondition analysis is both sound and adequate with respect to the operational semantics of the language. Using the induced expectation transformer, we provide formal analysis methods for the expected cost analysis and expected value analysis of classical-quantum programs. We illustrate the

1 INTRODUCTION

Quantum computation is a promising and emerging computational paradigm which can efficiently solve problems considered to be intractable on classical computers [Harrow et al. 2009; Shor 1999]. However, the unintuitive nature of quantum mechanics poses interesting and challenging questions for the design and analysis of quantum programming languages. Indeed, the quantum program dynamics are considerably more complicated compared to the behaviour of classical probabilistic programs. Therefore, formal reasoning about quantum programs requires the development of

QRAM PROGRAMS

Deriving Bounds on the Size of the Produced Circuits

Resource Analysis in QCDLs

Deriving Bounds on the Size of the Produced Circuits

Bounds Should be Parametric on the Input

Resource Analysis in QCDLs

Deriving Bounds on the Size of the Produced Circuits

Bounds Should be Parametric on the Input

- Deriving size bounds on just **one** circuit is a trivial problem.
- Parametric bounds, instead, allows us to derive, e.g., the **maximal problem size** achievable with a given amount of qubits.

Resource Analysis in QCDLs

Deriving Bounds on the Size of the Produced Circuits

Bounds Should be Parametric on the Input

How are Circuits Actually Built?

contributed articles

DOI:10.1145/2699415

The Quipper language offers a unified general-purpose programming framework for quantum computation.

BY BENOÎT VALIRON, NEIL J. ROSS, PETER SELINGER, D. SCOTT ALEXANDER, AND JONATHAN M. SMITH

Programming the Quantum Future

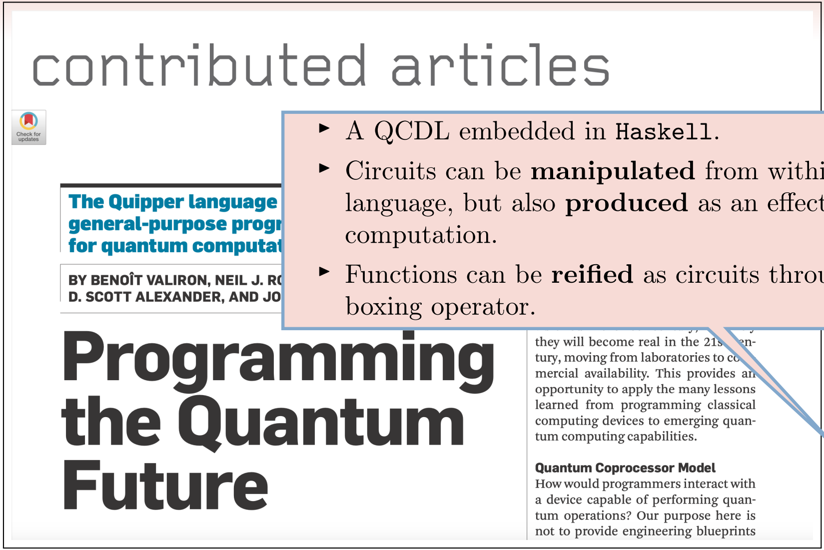
Quantum computation is a computing paradigm where data is encoded in the state of objects governed by the laws of quantum physics. Using quantum techniques, it is possible to design algorithms that outperform their best-known conventional, or classical, counterparts.

While quantum computers were envisioned in the 20th century, it is likely they will become real in the 21st century, moving from laboratories to commercial availability. This provides an opportunity to apply the many lessons learned from programming classical computing devices to emerging quantum computing capabilities.

Quantum Coprocessor Model

How would programmers interact with a device capable of performing quantum operations? Our purpose here is not to provide engineering blueprints

Quipper



contributed articles

Check for updates

The Quipper language general-purpose progr for quantum computat

BY BENOÎT VALIRON, NEIL J. RO D. SCOTT ALEXANDER, AND JO

Programming the Quantum Future

they will become real in the 21s en- tury, moving from laboratories to co mercial availability. This provides an opportunity to apply the many lessons learned from programming classical computing devices to emerging quan- tum computing capabilities.

Quantum Coprocessor Model

How would programmers interact with a device capable of performing quan- tum operations? Our purpose here is not to provide engineering blueprints

- A QCDL embedded in `Haskell`.
- Circuits can be **manipulated** from within the language, but also **produced** as an effect of any computation.
- Functions can be **reified** as circuits through a boxing operator.

`Quipper`

The Essence of Quipper

From Proto-Quipper ...

$$
\begin{array}{rl}
\textit{Terms} & M, N ::= \lambda x_A.M \mid \ell \mid \mathsf{box}_T\ M \mid (\vec{\ell}, C, \vec{r}) \mid \mathsf{apply}(M, N) \mid \cdots \\
\textit{Types} & A, B ::= \mathsf{Bit} \mid \mathsf{Qubit} \mid A \multimap B \mid \mathsf{Circ}(T, U) \mid \mathsf{List}\ A \mid \cdots
\end{array}
$$

The Essence of `Quipper`

From r . . .

A value encapsulating a circuit C

Terms	$M, N ::= \lambda x_A.M \mid \ell \mid \mathsf{box}_T\ M \mid (\vec{\ell}, C, \vec{r}) \mid \mathsf{apply}(M, N) \mid \cdots$
Types	$A, B ::= \mathsf{Bit} \mid \mathsf{Qubit} \mid A \multimap B \mid \mathsf{Circ}(T, U) \mid \mathsf{List}\ A \mid \cdots$

The Essence of `Quipper`

A value encapsulating a circuit C

Extends the current circuit with some new gates

From ... r

Terms	$M, N ::= \lambda x_A.M \mid \ell \mid \mathsf{box}_T\ M \mid (\vec{\ell}, C, \vec{r}) \mid \mathsf{apply}(M, N) \mid \cdots$
Types	$A, B ::= \mathsf{Bit} \mid \mathsf{Qubit} \mid A \multimap B \mid \mathsf{Circ}(T, U) \mid \mathsf{List}\ A \mid \cdots$

The Essence of `Quipper`

From … r

A value encapsulating a circuit C

Extends the current circuit with some new gates

$$
\begin{array}{lrcl}
\textit{Terms} & M, N & ::= & \lambda x_A.M \mid \ell \mid \mathsf{box}_T\ M \mid (\vec{\ell}, C, \vec{r}) \mid \mathsf{apply}(M, N) \mid \cdots \\
\textit{Types} & A, B & ::= & \mathsf{Bit} \mid \mathsf{Qubit} \mid A \multimap B \mid \mathsf{Circ}(T, U) \mid \mathsf{List}\ A \mid \cdots
\end{array}
$$

Type of circuits with input T and output U

The Essence of Quipper

From Proto-Quipper . . .

$$
\begin{array}{ll}
\textit{Terms} & M, N ::= \lambda x_A.M \mid \ell \mid \mathsf{box}_T\ M \mid (\vec{\ell}, C, \vec{r}) \mid \mathsf{apply}(M, N) \mid \cdots \\
\textit{Types} & A, B ::= \mathsf{Bit} \mid \mathsf{Qubit} \mid A \multimap B \mid \mathsf{Circ}(T, U) \mid \mathsf{List}\ A \mid \cdots
\end{array}
$$

. . . to Proto-Quipper-RA

$$
\begin{array}{ll}
\textit{Terms} & M, N ::= \lambda x_A.M \mid \ell \mid \mathsf{box}_T\ M \mid (\vec{\ell}, C, \vec{r}) \mid \mathsf{apply}(M, N) \mid \cdots \\
\textit{Types} & A, B ::= \mathsf{Bit} \mid \mathsf{Qubit} \mid A \multimap^I_T B \mid \mathsf{Circ}^I(T, U) \mid \mathsf{List}_{i \leq I} A \mid \cdots
\end{array}
$$

The Essence of Quipper

From Proto-Quipper ...

$$\begin{array}{rl} \textit{Terms} & M, N ::= \lambda x_A.M \mid \ell \mid \mathsf{box}_T\ M \mid (\vec{\ell}, C, \vec{r}) \mid \mathsf{apply}(M, N) \mid \cdots \\ \textit{Types} & A, B ::= \mathsf{Bit} \mid \mathsf{Qubit} \mid A \multimap B \mid \mathsf{Circ}(T, U) \mid \mathsf{List}\ A \mid \cdots \end{array}$$

... to Proto-Quipper-RA

$$\begin{array}{rl} \textit{Terms} & M, N ::= \lambda x_A.M \mid \ell \mid \mathsf{box}_T\ M \mid (\vec{\ell}, C, \vec{r}) \mid \mathsf{apply}(M, N) \mid \cdots \\ \textit{Types} & A, B ::= \mathsf{Bit} \mid \mathsf{Qubit} \mid A \multimap^I_T B \mid \mathsf{Circ}^I(T, U) \mid \mathsf{List}_{i \leq I} A \mid \cdots \end{array}$$

Circuits have a size.

The Essence of Quipper

From Proto-Quipper ...

$$\begin{array}{lrcl} \textit{Terms} & M, N & ::= & \lambda x_A.M \mid \ell \mid \mathsf{box}_T\ M \mid (\vec{\ell}, C, \vec{r}) \mid \mathsf{apply}(M, N) \mid \cdots \\ \textit{Types} & A, B & ::= & \mathsf{Bit} \mid \mathsf{Qubit} \mid A \multimap B \mid \mathsf{Circ}(T, U) \mid \mathsf{List}\ A \mid \cdots \end{array}$$

... to Proto-Quipper-RA

$$\begin{array}{lrcl} \textit{Terms} & M, N & ::= & \lambda x_A.M \mid \ell \mid \mathsf{box}_T\ M \mid (\vec{\ell}, C, \vec{r}) \mid \mathsf{apply}(M, N) \mid \cdots \\ \textit{Types} & A, B & ::= & \mathsf{Bit} \mid \mathsf{Qubit} \mid A \multimap^I_T B \mid \mathsf{Circ}^I(T, U) \mid \mathsf{List}_{i \leq I} A \mid \cdots \end{array}$$

Functions produce effects.

Circuits have a size.

Proto-Quipper-RA: Typing Judgements

$$\Theta \,;\, \Gamma \,;\, Q \vdash M : A \,;\, I$$

Proto-Quipper-RA: Typing Judgements

Term and type

$$\Theta \,;\, \Gamma \,;\, Q \vdash M : A \,;\, I$$

`Proto-Quipper-RA`: Typing Judgements

Term Variables

Term and type

$$\Theta\ ;\ \Gamma\ ;\ Q \vdash M : A\ ;\ I$$

Proto-Quipper-RA: Typing Judgements

Term Variables

Pointers to the circuit.

Term and type

$$\Theta\ ;\ \Gamma\ ;\ Q \vdash M : A\ ;\ I$$

Proto-Quipper-RA: Typing Judgements

Term Variables

Pointers to the circuit.

Term and type

$$\Theta \,;\, \Gamma \,;\, Q \vdash M : A \,;\, I$$

A parametric bound on the size of the underlying circuit

Proto-Quipper-RA: Typing Judgements

Term Variables

Pointers to the circuit.

Term and type

$$\Theta \; ; \; \Gamma \; ; \; Q \vdash M : A \; ; \; I$$

Parameters I depends on

A parametric bound on the size of the underlying circuit

`Proto-Quipper-RA`: Abstract Resource Operators

$$I \quad ::= \quad I \oplus J \mid I \circledast J \mid \cdots$$

Proto-Quipper-RA: Abstract Resource Operators

Size of **parallel** composition

$$I \ ::= \ I \oplus J \mid I \circledast J \mid \cdots$$

Proto-Quipper-RA:

Abst … R … ur … tors

Size of **parallel** composition

Size of **sequential** composition

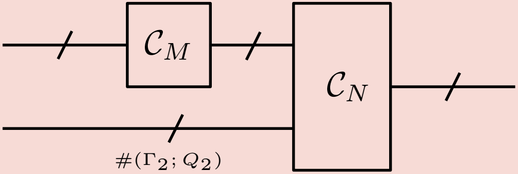
$$I \ ::= \ I \oplus J \mid I \circledast J \mid \cdots$$

Proto-Quipper-RA: Abstract Resource Operators

$$I \ ::= \ I \oplus J \mid I \circledast J \mid \cdots$$

$$\frac{\Theta; \Gamma_1; Q_1 \vdash M : A; I \quad \Theta; \Gamma_2, x : A; Q_2 \vdash N : B; J}{\Theta; \Gamma_1, \Gamma_2; Q_1, Q_2 \vdash \texttt{let}\ x = M\ \texttt{in}\ N : A; (I \oplus \#(\Gamma_2; Q_2)) \circledast J}$$

Proto-Quipper-RA: Abstract Resource Operators



$$\frac{\Theta; \Gamma_1; Q_1 \vdash M : A; I \qquad \Theta; \Gamma_2, x : A; Q_2 \vdash N : B; J}{\Theta; \Gamma_1, \Gamma_2; Q_1, Q_2 \vdash \texttt{let}\ x = M\ \texttt{in}\ N : A; (I \oplus \#(\Gamma_2; Q_2)) \circledast J}$$

Global vs. Local Metrics

Gate Count

Global vs. Local Metrics

- The **total number of gates** in the underlying circuit.
- Sometimes, you are interested in counting only **certain** gates (like binary gates, or swaps).

Gate Count

Global vs. Local Metrics

Gate Count

Width

Global vs. Local Metrics

- The **maximum amount of wires** active along the evaluation of the circuit.
- **Variations** thereof (e.g., only counting qubits).

Gate Count

Width

Global vs. Local Metrics

Gate Count

Width

Depth

Global vs. Local Metrics

Gate Count

Width

Depth

- The number of **parallel** evaluation steps.
- This is inherently **local**, and requires base types to be **labeled** with an index term I.

Main Result

Key Definitions

- A **resource metric** is a function from circuits to natural numbers capturing a given notion of size.
- A **resource metric interpretation** provides an interpretation of to the operators $\oplus, \circledast, \mathtt{e}, \ldots$
- A resource metric interpretation t is **coherent** with a resource metric μ if it satisfies a mild set of inequalities involving both t and μ.

Main Result

Key Definitions

- A **resource metric** is a function from circuits to natural numbers capturing a given notion of size.
- A **resource metric interpretation** provides an interpretation of to the operators $\oplus, \circledast, \mathtt{e}, \ldots$
- A resource metric interpretation t is **coherent** with a resource metric μ if it satisfies a mild set of inequalities involving both t and μ.

Correctness

Any instance of `Proto-Quipper-RA` induced by a resource metric interpretation which is coherent with μ derives judgments which are semantically sound with respect to μ.

A Tool

QuRA

A Tool

```
--- Quantum Fourier Transform with width and depth analysis
-- apply the controlled rotation gate to the target qubit trg
at iteration iter
let rotate = forall d. forall iter. lift forall step.
  \((ctrls, trg), ctrl)::((List[i<step] Qubit{d+iter+i+1},
Qubit{d+iter+step}), Qubit{d+iter+step}).
    let (ctrl, trg) = (force cr @(iter+1-step) @(d+iter+step)
@(d+iter+step)) ctrl trg in
    (ctrls:ctrl, trg) -- :: (List[i<step+1] Qubit{d+iter+i+1},
Qubit{d+iter+step+1})
in

-- apply the Quantum Fourier Transform to n qubits at depth d
let qft = lift forall n. forall d.
  \reg :: List[i<n] Qubit{d}.
    let qftIter = lift forall iter. -- define the iteration of
the QFT
      \(ctrls, trg)::(List[i<iter] Qubit{d+iter+i}, Qubit{d}).
        let revctrls = (force qrev @iter @d) ctrls in --
List[i<iter] Qubit{d+2*iter-(i+1)}
        let (ctrls, trg) = fold(rotate @d @iter, ([], trg),
revctrls) in
        -- note (ctrls, trg) :: (List[i<iter]
Qubit{d+iter+i+1}, Qubit{d+2*iter})
        let trg = (force hadamard @(d+2*iter)) trg in
        ctrls:trg -- List[i<iter+1] Qubit{d+iter+1+i}
    in fold(qftIter, [], reg) -- List[i<n] Qubit{d+n+i}
qft -- uncomment for generic, parametric bounds
--(force qft @4 @0) [(force qinit0), (force qinit0), (force
qinit0), (force qinit0)] -- uncomment for instance bounds
```

A Tool

```
$ qura examples/qft.pqr -g width -l depth

* Inferred type:
    !(forall n. forall d. List[i<n] Qubit{d} -o[n,0] List[i<n] Qubit{d+n+i})

* Inferred width upper bound: 0
```

Wrapping-Up

Take Home Messages

- Quantum computing shows great **promise** but faces **challenges** like decoherence, noise, and limited qubits.
- The task of deriving **resource bounds** for Quantum Circuit Description Languages (QCDLs) and for `Quipper` in particular turns out to be feasible, with the help of types.

Wrapping-Up

Take Home Messages

- Quantum computing shows great **promise** but faces **challenges** like decoherence, noise, and limited qubits.
- The task of deriving **resource bounds** for Quantum Circuit Description Languages (QCDLs) and for `Quipper` in particular turns out to be feasible, with the help of types.

Ongoing Work

- We can give an adequate **monadic** denotational semantics to the introduced type system, which systematizes the task of dealing with circuit metrics, and suggests **new ones**.

Wrapping-Up

Take Home Messages

- Quantum computing shows great **promise** but faces **challenges** like decoherence, noise, and limited qubits.
- The task of deriving **resource bounds** for Quantum Circuit Description Languages (QCDLs) and for `Quipper` in particular turns out to be feasible, with the help of types.

Ongoing Work

- We can give an adequate **monadic** denotational semantics to the introduced type system, which systematizes the task of dealing with circuit metrics, and suggests **new ones**.

Thank you! Questions?