ESOP 21
April 1st 2021

An Automated Deductive Verification Framework for Circuit-building Quantum Programs

Christophe Chareton (CEA/LIST, LMF)

Sébastien Bardin(CEA/LIST), François Bobot(CEA/LIST),

Valentin Perrelle (CEA/LIST) and Benoît Valiron (LMF)

Outline

A brief introduction to quantum computing

The challenge of validating quantum programs

- Focus on parametricity and automation

Automated Formal verification of quantum programs in QBRICKS

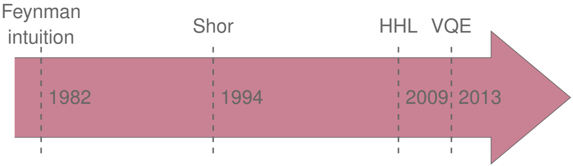
- A finely-tuned programming, spec and proof environment
- Dedicated first-order reasoning methods

Experimental evaluation

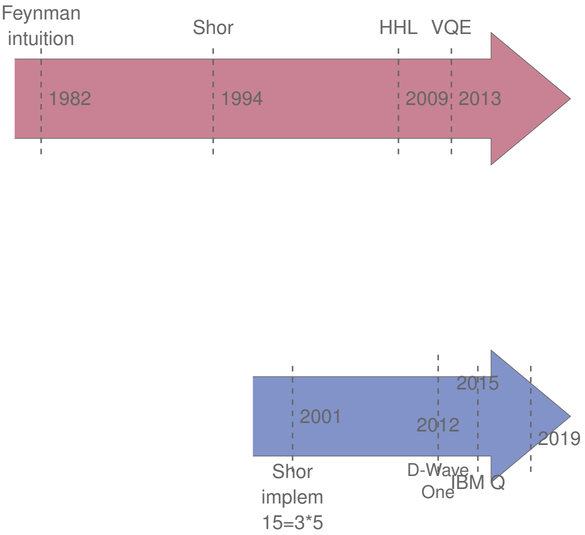
- Including the first proof of a Shor order-finding implementation

Conclusion

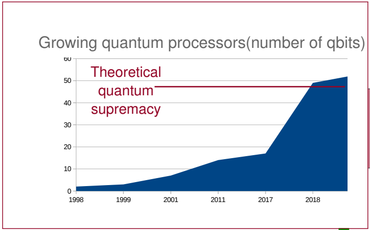
Quantum computing milestone history

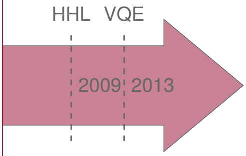


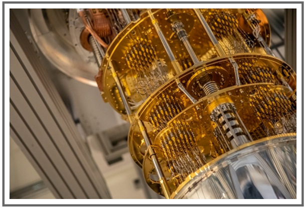
Quantum computing milestone history



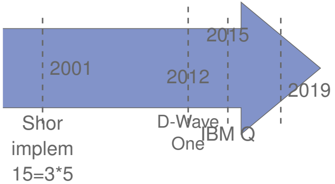
Quantum computing milestone history







TIME FOR SOFTWARE ENGINEERING !



Quantum memory state

- Classical world:

  🐱 **XOR** ~~🐱~~

- Quantum world:

  α_0 🐱 $\oplus$ α_1 ~~🐱~~

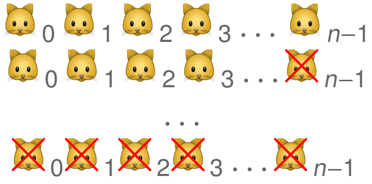
with amplitudes $\alpha_0, \alpha_1 \in \mathbb{C}$, $|\alpha_0|^2 + |\alpha_1|^2 = 1$,

Generalization of probabilities

Quantum memory state

- Quantum world: superposition over *n* qbits

$$|\alpha\rangle = \bigoplus \begin{matrix} \alpha_0 \\ \alpha_1 \\ \cdots \\ \alpha_{2^n-1} \end{matrix}$$



with $\sum_{j=0}^{2^n-1} \mid \alpha_j \mid^2 = 1$

$$|\alpha\rangle = \bigoplus{}^1 \begin{matrix} \alpha_0|0\rangle \\ \alpha_1|1\rangle \\ \cdots \\ \alpha_{2^n-1}|2^{n-1}\rangle \end{matrix} = \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \cdots \\ \alpha_{2^n-1} \end{pmatrix}$$

Quantum memory state

- Quantum world: superposition over *n* qbits

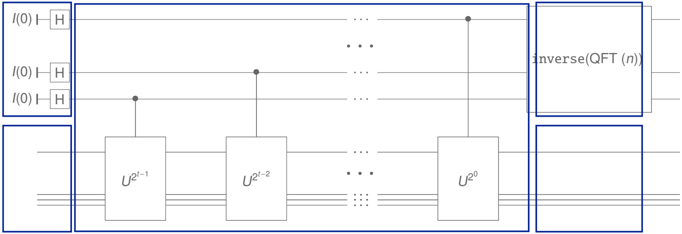
$|\alpha\rangle = \bigoplus$

+ Some strange rules:
  - no cloning
  - destructive measure
  - operations restricted to unitary (preserving norms)

    matrix interpretation $\texttt{Mat}(U)$ (of dimension $2^n \times 2^n$):

$$|\alpha\rangle \to \texttt{Mat}(U) \cdot |\alpha\rangle$$

In practice: quantum circuits

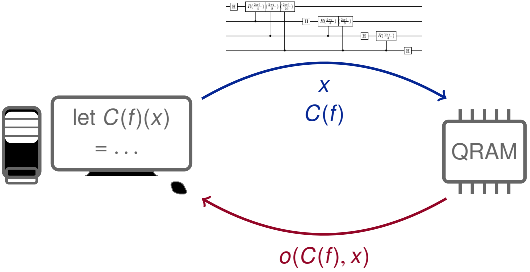


- a recursive set of elementary operations (**quantum gates**)
- unitary compositions :

sequence	→	matrix product
parallel	→	Kronecker product

The hybrid model

- A quantum co-processor (QRAM), controlled by a classical computer
  - Classical control flow
  - Quantum computing request, sent to the QRAM
- $\rightarrow$ Structured sequences of instructions: quantum circuits



A brief introduction to quantum computing

The challenge of validating quantum programs

Automated Formal verification of quantum programs in QBRICKS

Experimental evaluation

Conclusion

Algorithm specifications

Inputs: (1) A black-box $U_{x,n}$ which performs the transformation $|j\rangle|k\rangle \to |j\rangle|x^j k \bmod N\rangle$, for x co-prime to the $L-$bit number N, (2) $t = 2L + 1 + \lceil \log(2 + \frac{1}{2\epsilon})\rceil$ qubits initialized to $|0\rangle$, and (3) L qubits initialized to the state $|1\rangle$.

Outputs: The least integer $r > 0$ such that $x^r = 1 \pmod N$.

Runtime: $O(L^3)$ operations. Succeeds with probability $O(1)$.

Procedure:

1.	$|0\rangle|u\rangle$	initial state
2.	$\to \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|1\rangle$	create superposition
3.	$\to \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|x^j \bmod N\rangle$	apply $U_{x,N}$
	$\approx \frac{1}{\sqrt{r2^t}} \sum_{s=0}^{r-1} \sum_{j=0}^{2^t-1} e^{2\pi i s j/r} |j\rangle|u_s\rangle$	
4.	$\to \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |\widetilde{s/r}\rangle|u_s\rangle$	apply inverse Fourier transform to the first register
5.	$\to \widetilde{s/r}$	measure first register
6.	$\to r$	apply continued fractions algorithm

Shor-OF (from N & C, p. 232)

A specification preamble:

- **Input parameters (size, oracle, etc)**
- Functional correctness: Inputs-Outputs relation
- **Complexity**: number of elementary operations

Validation of quantum programs

Inputs: (1) A black-box $U_{x,n}$ which performs the transformation $|j\rangle|k\rangle \rightarrow |j\rangle|x^j k \bmod N\rangle$, for x co-prime to the L−bit number N, (2) $t = 2L + 1 + \lceil \log(2 + \frac{1}{2\epsilon}) \rceil$ qubits initialized to $|0\rangle$, and (3) L qubits initialized to the state $|1\rangle$.

Outputs: The least integer $r > 0$ such that $x^r = 1 \pmod N$.

Runtime: $O(L^3)$ operations. Succeeds with probability $O(1)$.

Procedure:

1. $|0\rangle|u\rangle$ initia…
2. $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|1\rangle$ crea…
3. $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|x^j \bmod N\rangle$

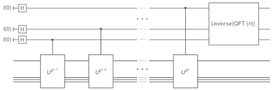
   $\approx \frac{1}{\sqrt{r2^t}} \sum_{s=0}^{r-1} \sum_{j=0}^{2^t-1} e^{2\pi i s j / r} |j\rangle|u…$
4. $\rightarrow \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |\widetilde{s/r}\rangle|u_s\rangle$ app… to th…
5. $\rightarrow |\widetilde{s/r}\rangle$ mea…
6. $\rightarrow r$ app… algo…

```
qft_internal :: [Qubit] -> Circ [Qubit]
qft_internal [] = return []
qft_internal [x] = do
  hadamard x
  return [x]
qft_internal (x:xs) = do
  xs' <- qft_internal xs
  xs'' <- rotations x xs' (length xs')
  x' <- hadamard x
  return (x':xs'')
  where
    -- Auxiliary function used by 'qft'.
    rotations :: Qubit -> [Qubit] -> Int -> Circ [Qubit]
    rotations _ [] _ = return []
    rotations c (q:qs) n = do
      qs' <- rotations c qs n
      q' <- rGate ((n + 1) - length qs) q `controlled` c
      return (q':qs')
```

Quipper QFT circuit building function

A specification preamble:

- **Input parameters (size, oracle, etc)**
- Functional correctness: Inputs-Outputs relation
- **Complexity**: number of elementary operations



Validation of quantum programs

- Specifications: the circuit should meet the spec **for any value of parameters**
- Quantum programming is non-intuitive
  → High risk for bugs!

Inputs: (1) A black-box $U_{x,n}$ which performs the transfo
$|j\rangle|k\rangle \to |j\rangle|x^j k \bmod N\rangle$, for x co-prime to the L−bit num
(2) $t = 2L + 1 + \lceil \log(2 + \frac{1}{2\epsilon}) \rceil$ qubits initialized to $|0\rangle$, and
(3) L qubits initialized to the state $|1\rangle$.

Outputs: The least integer $r > 0$ such that $x^r = 1$ (mod

Runtime: $O(L^3)$ operations. Succeeds with probability

Procedure:

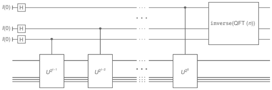
1. $|0\rangle|u\rangle$ initia
2. $\to \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|1\rangle$ crea
3. $\to \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|x^j \bmod N\rangle$
   $\approx \frac{1}{\sqrt{r2^t}} \sum_{s=0}^{r-1} \sum_{j=0}^{2^t-1} e^{2\pi i s j/r} |j\rangle|u$
4. $\to \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |\widetilde{s/r}\rangle|u_s\rangle$ appl to th
5. $\to |\widetilde{s/r}\rangle$ mea
6. $\to r$ appl algo

```
qft_internal
qft_internal
qft_internal [x] = do
  hadamard x
  return [x]
qft_internal (x:xs) = do
  xs' <- qft_internal xs
  xs'' <- rotations x xs' (length xs')
  x' <- hadamard x
  return (x':xs'')
  where
    -- Auxiliary function used by 'qft'.
    rotations :: Qubit -> [Qubit] -> Int -> Circ [Qubit]
    rotations _ [] _ = return []
    rotations c (q:qs) n = do
      qs' <- rotations c qs n
      q' <- rGate ((n + 1) - length qs) q `controlled` c
      return (q':qs')
```

Quipper QFT circuit building function

A specification preamble:

- **Input parameters (size, oracle, etc)**
- Functional correctness: Inputs-Outputs relation
- **Complexity**: number of elementary operations



Validation of quantum programs

- Specifications: the circuit should meet the spec **for any value of parameters**
- Quantum programming is non-intuitive
  $\rightarrow$ High risk for bugs!

Inputs: (1) A black-box $U_{x,n}$ which performs the transfo
$|j\rangle|k\rangle \rightarrow |j\rangle|x^j k \bmod N\rangle$, for x co-prime to the L–bit num
(2) $t = 2L + 1 + \lceil \log(2 + \frac{1}{2\epsilon}) \rceil$ qubits initialized to $|0\rangle$, and
(3) L qubits initialized to the state $|1\rangle$.

Outputs: The least integer $r > 0$ such that $x^r = 1$ (mod

Runtime: $O(L^3)$ operations. Succeeds with probability

Procedure:

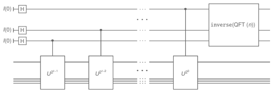
1. $|0\rangle|u\rangle$ initia
2. $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|1\rangle$ crea
3. $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|x^j \bmod N\rangle$
   $\approx \frac{1}{\sqrt{r2^t}} \sum_{s=0}^{r-1} \sum_{j=0}^{2^t-1} e^{2\pi i s j/r}|j\rangle|u$
4. $\rightarrow \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |\widetilde{s/r}\rangle|u_s\rangle$ app to th
5. $\rightarrow \widetilde{s/r}$ mea
6. $\rightarrow r$ app algo

```
qft_internal
qft_internal
qft_internal [x] = do
  hadamard x
  return [x]
qft_internal (x:xs) = do
  xs' <- qft_internal xs
  xs'' <- rotations x xs' (length xs')
  x' <- hadamard x
  return (x':xs'')
  where
    -- Auxiliary function used by 'qft'.
    rotations :: Qubit -> [Qubit] -> Int -> Circ [Qubit]
    rotations _ [] _ = return []
...gth qs) q `controlled` c
```

circuit building function

What about testing?

- No means to control an execution
- **Tests are expensive and often statistical**



Validation of quantum programs

- Specifications: the circuit should meet the spec **for any value of parameters**
- Quantum programming is non-intuitive
  → High risk for bugs!

Inputs: (1) A black-box $U_{x,n}$ which performs the transfo… $|j\rangle|k\rangle \rightarrow |j\rangle|x^j k \bmod N\rangle$, for x co-prime to the L-bit num…
(2) $t = 2L + 1 + \lceil \log(2 + \frac{1}{2\epsilon}) \rceil$ qubits initialized to $|0\rangle$, and…
(3) L qubits initialized to the state $|1\rangle$.

Outputs: The least integer $r > 0$ such that $x^r = 1$ (mod…

Runtime: $O(L^3)$ operations. Succeeds with probability…

Procedure:

1. $|0\rangle|u\rangle$
2. $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|1\rangle$
3. $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|x^j \bmod N\rangle$
   $\approx \frac{1}{\sqrt{r2^t}} \sum_{s=0}^{r-1} \sum_{j=0}^{2^t-1} e^{2\pi i s j/r} |j\rangle|u…$
4. $\rightarrow \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |\widetilde{s/r}\rangle|u_s\rangle$
5. $\rightarrow |\widetilde{s/r}\rangle$
6. $\rightarrow r$

```
qft_internal
qft_internal
qft_internal [x] = do
  hadamard x
  return [x]
qft_internal (x:xs) = do
  xs' <- qft_internal xs
  xs'' <- rotations x xs' (length xs')
  x' <- hadamard x
  return (x':xs'')
  where
   -- Auxiliary function used by 'qft'.
   rotations :: Qubit -> [Qubit] -> Int -> Circ [Qubit]
   rotations _ [] _ = return []
```

What about testing?

- No means to control an execution
- **Tests are expensive and often statistical**

What about full verification?

- No need to execute
- unbounded state space
- absolute guarantee

université PARIS-SACLAY

Challenges

- double layer programming paradigm: higher-order functions
- non standard theories (probabilities, complex numbers, kronecker product, etc)

Challenges

- double layer programming paradigm: higher-order functions
- non standard theories (probabilities, complex numbers, kronecker product, etc)
- Additional requirements:
  - parametrized programming
  - automated proof support
  - complexity specifications

Deductive verification approach

Prior works

	Circuit	Parametrized	Proof automation	Complexity specifications
Path-sums	✔	✗	✔	✗
SQIR(Coq)	✔	✔	✗	could
QHL(Isabelle/HOL)	✗	✔	✗	✗

Our approach

	Circuit	Parametrized	Proof automation	Complexity specifications
Path-sums	✔	✗	✔	✗
SQIR(Coq)	✔	✔	✗	could
QHL(Isabelle/HOL)	✗	✔	✗	✗

Trade-off automation Vs parametricity (higher-order reasoning)

Our approach

	Circuit	Parametrized	Proof automation	Complexity specifications
Path-sums	✓	✗	✓	✗
SQIR(Coq)	✓	✓	✗	could
QHL(Isabelle/HOL)	✗	✓	✗	✗

Trade-off automation Vs parametricity (higher-order reasoning)

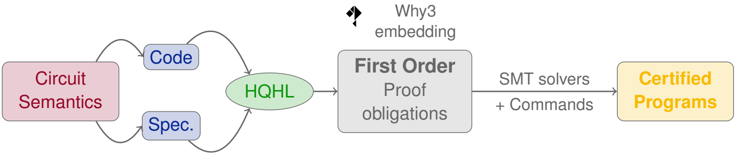
QBRICKS (Why3)	✓	✓	↗	✓

- first-order reasoning
- adequate deduction rules

QBRICKS contributions

- A programming, spec and proof framework, QBRICKS-DSL + QBRICKS-SPEC + HQHL, yielding **first-order proof obligations**
- A flexible symbolic representation for **first order reasoning** about quantum states and transformations: PPS
- Dedicated mathematical libraries, +14 kLoC
- Non trivial case studies (Shor-OF, Grover, QPE, ,etc)

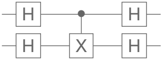
→ Publicly available at `https://cchareton.github.io/Qbricks/`



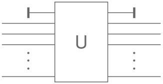
Programming framework: QBRICKS-DSL

- A minimal set of primary functions
  - elementary gates

  - compositions: parallel/sequence



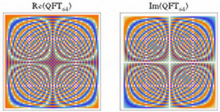
  - ancilla creation/anihilation



- derived high level combinators : inversion, control, qbit permutations, etc
- **bounded** iterations

Symbolic representation: path-sums

Matrix semantics: QFT(64):



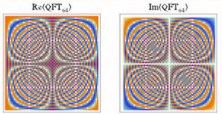
$\vert 0\rangle\vert u\rangle$	initial state
$\rightarrow \frac{1}{\sqrt{2^t}}\sum_{j=0}^{2^t-1}\vert j\rangle\vert 1\rangle$	create superposition
$\rightarrow \frac{1}{\sqrt{2^t}}\sum_{j=0}^{2^t-1}\vert j\rangle\vert x^j \bmod N\rangle$	apply $U_{x,N}$
$\approx \frac{1}{\sqrt{r2^t}}\sum_{s=0}^{r-1}\sum_{j=0}^{2^t-1} e^{2\pi i s j/r}\vert j\rangle\vert u_s\rangle$	
$\rightarrow \frac{1}{\sqrt{r}}\sum_{s=0}^{r-1}\vert\widetilde{s/r}\rangle\vert u_s\rangle$	apply inverse Fourier transform to the first register
$\rightarrow \vert\widetilde{s/r}\rangle$	measure first register

$$\rightarrow \frac{1}{\sqrt{2^t}}\sum_{j=0}^{2^t-1}|j\rangle|x^j \bmod N\rangle$$

$$\approx \frac{1}{\sqrt{r2^t}}\sum_{s=0}^{r-1}\sum_{j=0}^{2^t-1} e^{2\pi i s j/r}|j\rangle|u_s\rangle$$

Symbolic representation: path-sums

Matrix semantics: QFT(64):



$\|0\rangle\|u\rangle$	initial state
$\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} \|j\rangle\|1\rangle$	create superposition
$\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} \|j\rangle\|x^j \bmod N\rangle$	apply $U_{x,N}$
$\approx \frac{1}{\sqrt{r2^t}} \sum_{s=0}^{r-1} \sum_{j=0}^{2^t-1} e^{2\pi i s j/r}\|j\rangle\|u_s\rangle$	
$\rightarrow \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} \widetilde{\|s/r\rangle}\|u_s\rangle$	apply inverse Fourier transform to the first register
$\rightarrow \widetilde{\|s/r\rangle}$	measure first register

$$\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|x^j \bmod N\rangle$$

$$\approx \frac{1}{\sqrt{r2^t}} \sum_{s=0}^{r-1} \sum_{j=0}^{2^t-1} e^{2\pi i s j/r}|j\rangle|u_s\rangle$$

Path-sum semantics[Amy 2019]: build language term

$$\mathrm{PS} : |k\rangle \rightarrow \frac{1}{\sqrt{2^r}} \sum_{j=0}^{2^{r-1}} e^{2\cdot\pi\cdot f(k,j)}|g(k,j)\rangle$$

with instances of language variables:

- r: `int`
- f : `bitvec` → `bitvec` → `complex`
- g : `bitvec` → `bitvec` → `bitvec`

- Concise in practice
- Good compositional properties
- But no parametricity

Extension: parametrized path-sums

Parametrized circuits $C(\overrightarrow{p})$:

$$\mathrm{PPS}_{C,\overrightarrow{p}} : |k\rangle \to \frac{1}{\sqrt{2^{r_{C,\overrightarrow{p}}}}} \sum_{j=0}^{2^{r_{C,\overrightarrow{p}}}-1} e^{2\cdot\pi\cdot f_{C,\overrightarrow{p}}(k,j)} |g_{C,\overrightarrow{p}}(k,j)\rangle$$

- simple first-order modular reasoning upon each component $r_{C,\overrightarrow{p}}, f_{C,\overrightarrow{p}}, g_{C,\overrightarrow{p}}$
- **simple-first order composition rules**

→ at use for both spec (QBRICKS-SPEC) and deduction (HQHL)

Dedicated mathematical libraries

	Lines of code	Lemmas	Modules	Definitions
Mathematics libraries	14695	1614	77	328
Sets	532	59	4	14
Algebra	2091	190	10	37
Arithmetics	538	77	4	7
Binary arithmetics	1778	189	8	42
Complex numbers	2226	344	15	57
Quantum data	3335	310	12	68
Exponentiation	843	100	4	4
Iterators	861	72	6	30
Functions	259	33	3	8
Kronecker product	420	41	2	8
Unity circle	1812	199	9	53

A brief introduction to quantum computing

The challenge of validating quantum programs

Automated Formal verification of quantum programs in QBRICKS

Experimental evaluation

Conclusion

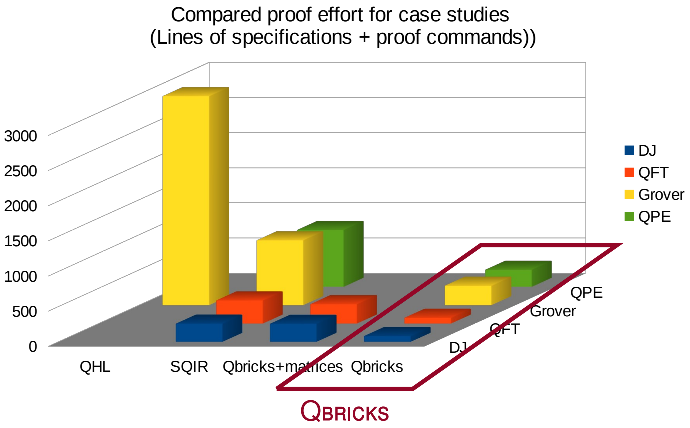
Experimental evaluation

First ever certified implementation of Shor-OF

	LoC + Spec	POs	Automation		#Cmd
			# Aut.	% Aut.	
Grover	193	505	479	>94%	125
QFT	65	62	53	>85%	37
QPE	175	282	262	>92%	94
Shor-OF	923	2473	2386	>96%	421
Shor-OF (full)	1163	2817	2701	>95%	552
Total	**1423**	**3394**	3241	**>95%**	**716**

#Aut.: automatically proven POs — #Cmd: interactive commands

Compared experimental evaluation



A brief introduction to quantum computing

The challenge of validating quantum programs

Automated Formal verification of quantum programs in QBRICKS

Experimental evaluation

Conclusion

Conclusion

- QBRICKS: a framework for formally verified circuit-building quantum programs, featuring **parametricity** and **automation**
- Non trivial case studies, **Shor-OF** first ever verified publication
- **PPS** as a tool of choice for quantum verification
- **Quantum-oriented math. libraries** in Why3

`https://cchareton.github.io/Qbricks/`

Commissariat à l'énergie atomique et aux énergies alternatives
CEA Tech List
Centre de Saclay — 91191 Gif-sur-Yvette Cedex
www-list.cea.fr

Syntax for Qbricks-DSL

Expression	e	$::=$	$x \mid c \mid f(e_1,\dots,e_n) \mid \texttt{let}\ \langle x_1,\dots,x_n\rangle = e\ \texttt{in}\ e' \mid$ $\texttt{if}\ e_1\ \texttt{then}\ e_2\ \texttt{else}\ e_3 \mid \texttt{iter}\ f\ e_1\ e_2$
Data Constructor	c	$::=$	$\underline{n} \mid \texttt{tt} \mid \texttt{ff} \mid \langle e_1,\dots,e_n\rangle \mid \texttt{CNOT} \mid \texttt{SWAP} \mid \texttt{ID} \mid \texttt{H} \mid \texttt{Ph}(e) \mid \texttt{R}_z(e) \mid$ $\texttt{ANC}(e) \mid \texttt{SEQ}(e_1,e_2) \mid \texttt{PAR}(e_1,e_2)$
Function	f	$::=$	$f_{\mathrm{d}} \mid f_{\mathrm{c}}$
Declaration	d	$::=$	$\texttt{let}\ f_{\mathrm{d}}(x_1,\dots,x_n) = e$
Type	A	$::=$	$\texttt{bool} \mid \texttt{int} \mid \top \mid A_1 \times \cdots \times A_n \mid \texttt{circ}.$
Value	v	$::=$	$x \mid \underline{n} \mid \texttt{tt} \mid \texttt{ff} \mid \langle v_1,\dots,v_n\rangle \mid$ $\texttt{CNOT} \mid \texttt{SWAP} \mid \texttt{ID} \mid \texttt{H} \mid \texttt{Ph}(\underline{n}) \mid \texttt{R}_z(\underline{n}) \mid \texttt{ANC}(v) \mid \texttt{SEQ}(v_1,v_2) \mid \texttt{PAR}(v_1,v_2)$
Context	$C[-]$	$::=$	$[-] \mid f(v_1,\dots v_{i-1}, C[-], e_{i+1},\dots,e_n) \mid$ $\texttt{let}\ \langle x_1,\dots,x_n\rangle = C[-]\ \texttt{in}\ e' \mid \texttt{if}\ C[-]\ \texttt{then}\ e_2\ \texttt{else}\ e_3 \mid$ $\texttt{iter}\ f\ C[-]\ e \mid \texttt{iter}\ f\ v\ C[-] \mid \langle v_1,\dots v_{i-1}, C[-], e_{i+1},\dots,e_n\rangle \mid$ $\texttt{CNOT} \mid \texttt{ID} \mid \texttt{H} \mid \texttt{Ph}(C[-]) \mid \texttt{R}_z(C[-]) \mid \texttt{ANC}(C[-]) \mid$ $\texttt{SEQ}(C[-],e) \mid \texttt{SEQ}(v,C[-]) \mid \texttt{PAR}(C[-],e) \mid \texttt{PAR}(v,C[-])$

Typing rules for QBRICKS-DSL

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{\Gamma \vdash f : A_1 \times \cdots \times A_n \to B \qquad \Gamma \vdash e_i : A_i}{\Gamma \vdash f(e_1, \ldots, e_n) : B}$$

$$\frac{\Gamma \vdash e_i : A_i}{\Gamma \vdash \langle e_1, \ldots, e_n \rangle : A_1 \times \cdots \times A_n}$$

$$\frac{\Gamma \vdash e_1 : A_1 \times \cdots \times A_n \qquad \Gamma, x_1 : A_1, \ldots, x_n : A_n \vdash e_2 : B}{\Gamma \vdash \texttt{let}\, \langle x_1, \ldots, x_n \rangle = e_1 \,\texttt{in}\, e_2 : B}$$

$$\frac{\Gamma \vdash e_1 : \texttt{bool} \qquad \Gamma \vdash e_2 : A \qquad \Gamma \vdash e_3 : A}{\Gamma \vdash \texttt{if}\, e_1 \,\texttt{then}\, e_2 \,\texttt{else}\, e_3 : A}$$

$$\frac{f : A \to A \qquad \Gamma \vdash e_1 : \texttt{int} \qquad \Gamma \vdash e_2 : A}{\Gamma \vdash \texttt{iter}\, f\ e_1\ e_2 : A}$$

Deduction rules for QBRICKS

HQHL rules for term constructs

$$\frac{\Gamma, x \Vdash \{\phi \wedge x \leq 0\}\ e_2\ \{P[x, \texttt{result}]\} \quad \Gamma, x, y \Vdash \{\phi \wedge P[x,y]\}\ f(y)\ \{P[x+1, \texttt{result}]\}}{\Gamma \Vdash \{\phi\}\ \texttt{iter}\ f\ \hat{e}_1\ e_2\ \{P[\hat{e}_1, \texttt{result}]\}}\ (\texttt{iter})$$

$$\frac{\Gamma \Vdash \{P\}e_1\{Q[x_i := \texttt{result}_i\} \quad \Gamma, x_1, \ldots, x_n \Vdash \{Q\}e_2\{R\}}{\Gamma \Vdash \{P\}\texttt{let}\ x_1, \ldots, x_n = e_1\ \texttt{in}\ e_2\{R\}}\ (\texttt{let})$$

$$\frac{\Gamma \Vdash \{P\}e_1\{Q[x := \texttt{result}\} \quad \Gamma, x \Vdash \{Q \wedge x\}e_2\{R\} \quad \Gamma, x \Vdash \{Q \wedge \neg x\}e_3\{R\}}{\Gamma \Vdash \{P\}\texttt{if}\ e_1\ \texttt{then}\ e_2[x := e_1]\ \texttt{else}\ e_3[x := e_1]\{R\}}\ (\texttt{if})$$

$$\frac{\forall i, \Gamma \Vdash \{P\}e_i\{R_i[\texttt{result}]\}}{\Gamma \Vdash \{P\}\langle e_1, \ldots, e_2\rangle\{R_1[\texttt{result}_1] \wedge \cdots \wedge R_n[\texttt{result}_n]\}}\ (\texttt{tuple})$$

$$\frac{f(x_1, \ldots, x_n) \triangleq e \quad \Gamma \Vdash \{P\}e[x_1 := e_1, \ldots, x_n := e_n]\{R\}}{\Gamma \Vdash \{P\}f(e_1, \ldots, e_n)\{R\}}\ (\texttt{decl})$$

$$\frac{\Gamma \vdash P \rightarrow P' \quad \Gamma \Vdash \{P'\}\ e\ \{Q'\} : A \quad \Gamma, \texttt{result} : A \vdash Q' \rightarrow Q}{\Gamma \Vdash \{P\}e\{Q\} : A}\ (\texttt{weaken})$$

$$\frac{\Gamma \vdash e_1 = e_2 : A \quad \Gamma \Vdash \{P[e_1]\}\ e[e_1]\ \{Q[e_1]\} : A}{\Gamma \Vdash \{P[e_2]\}\ e[e_2]\ \{Q[e_2]\} : A}\ (\texttt{eq})$$

Deduction rules for sequence of circuits

$$\frac{\texttt{Prec-SEQ} \quad \triangleq \quad \begin{array}{l} \Gamma \Vdash \{\phi\}C_1\{(\texttt{result}, \{p\}) = w\} \\ \Gamma \Vdash \{\phi\}C_1\{(\texttt{result}, \{p\}) = w\} \end{array}}{\Gamma \Vdash \{\phi\}\texttt{SEQ}(C_1, C_2)\{(\texttt{result}, \{p\}) = w\}} \ \texttt{SEQ}_w$$

$$\frac{\{\texttt{Prec-SEQ}\} \quad \begin{array}{lll} \Gamma & \Vdash & \{\phi_1\}C_1\{(\texttt{result}, \{p\}) = r_1(\{p\})\} \\ \Gamma & \Vdash & \{\phi_2\}C_2\{(\texttt{result}, \{p\}) = r_2(\{p\})\} \end{array}}{\Gamma \Vdash \{\phi_1 \ \wedge \ \phi_2\}\texttt{SEQ}(C_1, C_2)\{(\texttt{result}, \{p\}) = r_1(\{p\}) + r_2(\{p\})\}} \ \texttt{SEQ}_r$$

$$\frac{\{\texttt{Prec-SEQ}\} \quad \begin{array}{lll} \Gamma & \Vdash & \{\phi_1\}C_1\{(\texttt{result}, \{p\})(x, y_1) = a_1(\{p\}, x, y_1)\} \\ \Gamma & \Vdash & \{\phi_1\}C_1\{(\texttt{result}, \{p\})(x, y_1) = k_1(\{p\}, x, y_1)\} \\ \Gamma & \Vdash & \{\phi_2\}C_2\{(\texttt{result}, \{p\})(k_1(\{p\}, x, y_1), y_2) \\ & & = a_2(\{p\}, x, y_1, y_2)\} \end{array}}{\begin{array}{l} \Gamma \Vdash \{\phi_1 \ \wedge \ \phi_2\}\texttt{SEQ}(C_1, C_2)\{(\texttt{result}, \{p\})(x, y_1 \cdot y_2) \\ \quad = a_1(\{p\}, x, y_1) + a_2(\{p\}, x, y_1, y_2)\} \end{array}} \ \texttt{SEQ}_a$$

$$\frac{\{\texttt{Prec-SEQ}\} \quad \begin{array}{lll} \Gamma & \Vdash & \{\phi_1\}C_1\{(\texttt{result}, \{p\})(x, y_1) = k_1(\{p\}, x, y_1)\} \\ \Gamma & \Vdash & \{\phi_2\}C_2\{(\texttt{result}, \{p\})(k_1(\{p\}, x, y_1), y_2) \\ & & = k_2(\{p\}, x, y_1 \cdot y_2)\} \end{array}}{\Gamma \Vdash \{\phi_1 \ \wedge \ \phi_2\}\texttt{SEQ}(C_1, C_2)\{(\texttt{result}, \{p\})(x, y_1 \cdot y_2) = k_2(\{p\}, x, y_1 \cdot y_2)\}} \ \texttt{SEQ}_k$$

A dedicated proof engine

PPS → **first order reasoning**

COMPOSITIONAL FOL REASONING.(CIRCUIT SEQUENTIAL COMPOSITION: PHASE)

$$\frac{\Gamma \quad \vdash \quad \texttt{width}(\texttt{D}(\vec{p})) \ = \ \texttt{width}(\texttt{E}(\vec{p}))}{\Gamma, (k, j_1, j_2) : \texttt{int} \quad \vdash \quad \texttt{f}_{\texttt{seq(D,E)},\vec{p}} = \texttt{f}_{\texttt{D},\vec{p}}(k, j_1) + \texttt{f}_{\texttt{E},\vec{p}}(\texttt{g}_{\texttt{D},\vec{p}}(k, j_1), j_2)}$$

CIRCUIT APPLICATION:

$$\frac{\Gamma \vdash \texttt{basis\_ket}(|k\rangle)}{\Gamma \vdash \texttt{Output}(C(\vec{p}) \ ,|k\rangle) = \mathrm{PS}(r_{C,\vec{p}}, f_{C,\vec{p}}, g_{C,\vec{p}})(|k\rangle)}$$

MEASURE OUTPUT PROBABILITY:

$$\frac{\Gamma \vdash \texttt{basis\_ket}(|j\rangle)}{\texttt{proba\_measure}(C(\vec{p}), |j\rangle, k) = \frac{1}{\sqrt{2^{r_{C,\vec{p}}}}} \mid \sum_{k=0 \ g_{C,\vec{p}}(j,k)=\texttt{bv}(i)}^{r_{C,\vec{p}}} f_{C,\vec{p}}(j, k) \mid^2}$$

Metrics about Qbricks implementation

	Lines of code	Lemmas	Modules	Definitions
Mathematics libraries	14695	1614	77	328
Sets	532	59	4	14
Algebra	2091	190	10	37
Arithmetics	538	77	4	7
Binary arithmetics	1778	189	8	42
Complex numbers	2226	344	15	57
Quantum data	3335	310	12	68
Exponentiation	843	100	4	4
Iterators	861	72	6	30
Functions	259	33	3	8
Kronecker product	420	41	2	8
Unity circle	1812	199	9	53
Qbricks core	1357	50	5	35
Semantics reasoning	744	55	3	35
Generic functions	517	12	5	34
Total	17313	1731	78	410

Compared case studies

	QBRICKS hops				QBRICKS Matrix			
	LoC	Spec	Cmd	Spec+Cmd	LoC	Spec	Cmd	Spec+Cmd
DJ	11	46	45	91	11	129	131(>2.9x)	260(>2.8x)
QFT	18	47	37	84	18	172	106(>2.8x)	278 (>3.3x)
Grover	42	293	123	416				
QPE	33	179	141	320				

	Sqir				QHL			
	LoC	Spec	Cmd	Spec+Cmd	LoC	Spec	Cmd	Spec+Cmd
DJ	10	39	222(>4.9x)	261(>2.8x)				
QFT	10	44	287(>7.7x)	331(>3.9x)				
Grover	15	121	805(>6.5x)	926(>2.2x)	90	1263	1712(>13.9x)	2975 (>7.1x)
QPE	40	86	726(>5.1x)	812(>2.5x)				

#LoC.: lines of code — # Spec.: lines of specifications and lemmas #Cmd: proof commands

Experimental evaluation (details)

Non trivial case studies (Grover, QPE, Shor-OF,etc)

→ **first ever certified implementation of Shor-OF**

	#LoC + Spec	#Def.	#Lem.	#POs	Automation		#Cmd
					# Aut.	% Aut.	
Grover	193	6	8	505	479	>94%	125
QFT	65	3	0	62	53	>85%	37
QPE	175	3	8	282	262	>92%	94
Shor-OF	923	28	14	2473	2386	>96%	421
Shor-OF (full)	1163	34	22	2817	2701	>95%	552
Total	**1423**	42	31	**3394**	3241	**>95%**	**716**

#LoC + Spec.: lines of decorated code — #Aut.: automatically proven POs — #Cmd: interactive commands