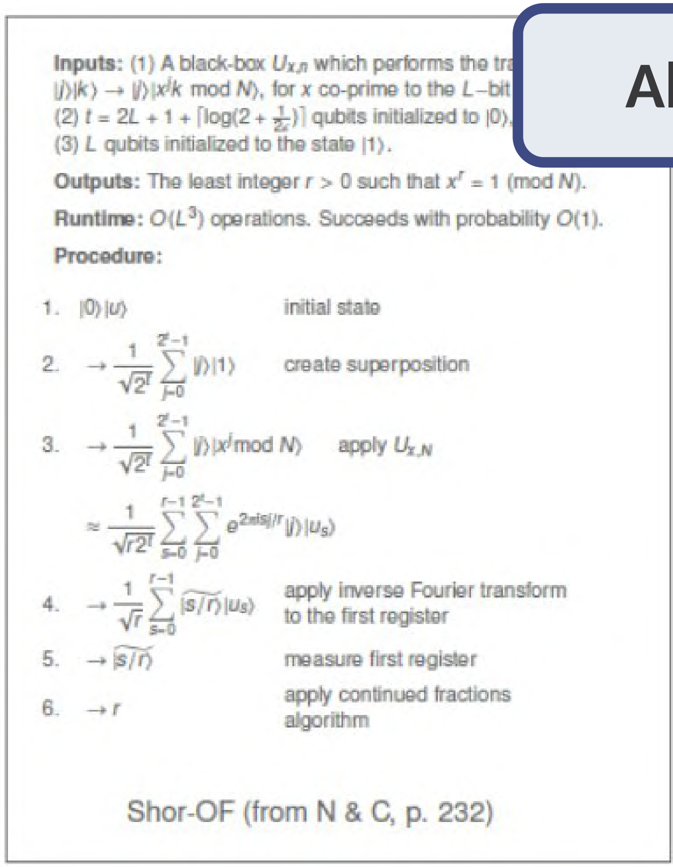
Quantum programming and automatic code analysis

Christophe Chareton, Sébastien Bardin

Quantum computing : our challenge



Inputs: (1) A black-box $U_{x,n}$ which performs the tra
$|j\rangle|k\rangle \rightarrow |j\rangle|x^j k \bmod N\rangle$, for x co-prime to the L–bit
(2) $t = 2L + 1 + \lceil \log(2 + \frac{1}{2\epsilon}) \rceil$ qubits initialized to $|0\rangle$,
(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. $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|1\rangle$ create superposition
3. $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|x^j \bmod N\rangle$ apply $U_{x,N}$
   $\approx \frac{1}{\sqrt{r2^t}} \sum_{s=0}^{r-1} \sum_{j=0}^{2^t-1} e^{2\pi i s j/r} |j\rangle|u_s\rangle$
4. $\rightarrow \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |\widetilde{s/r}\rangle|u_s\rangle$ apply inverse Fourier transform to the first register
5. $\rightarrow \widetilde{s/r}$ measure first register
6. $\rightarrow r$ apply continued fractions algorithm

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

Algos

How?

Hardware

08/06/2023

Quantum programming and formal verification

1. **Quantum programming, program specifications and verification**
2. Main challenges
3. A word about our works

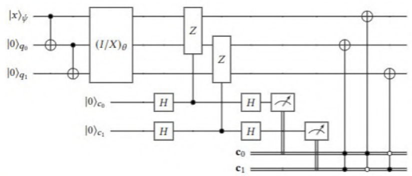
The hybrid model

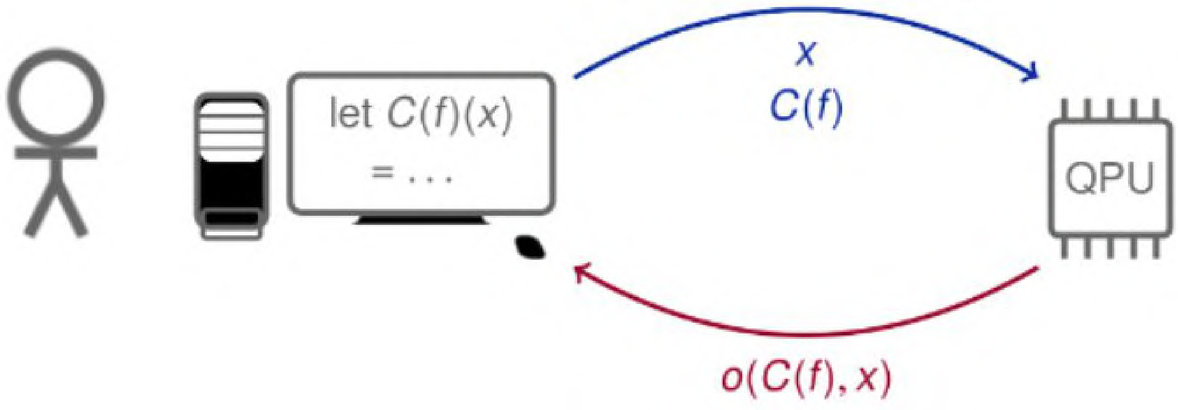
A quantum co-processor (QPU), controlled by a classical computer

- classical control flow
- CPU ⇒ QPU : quantum computing requests, sent to the QPU

→structured sequenced of instructions: **quantum circuits**

- QPU ⇒ CPU: **probabilistic** computation results (**classical** information)





Verification : specifications

Algorithm: Quantum order-finding

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

A specification preamble:

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

Verification : specifications

Algorithm: Quantum order-finding

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. $|0\rangle|1\rangle$ initial state
2. $\to \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|1\rangle$
3. $\to \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|x^j \bmod N\rangle$

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

A specification preamble:

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

Adequate implementation should come with evidence regarding the specs

Verification : specifications

Algorithm: Quantum order-finding

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) $t = 2L + 1 + \lceil \log\left(2 + \frac{1}{2\epsilon}\right) \rceil$ qubits initialized to $|0\rangle$, and (3) L qubits initiali to the state $|1\rangle$.

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

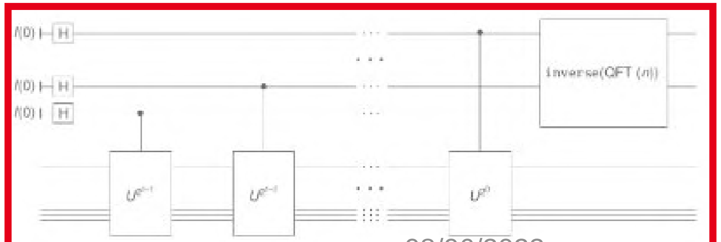
Runtime: $O(L^3)$ operations. Succeeds with probability $O(1)$.

Procedure:

1. $|0\rangle|1\rangle$ — 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 register
5. $\to \widetilde{s/r}$ — measure first register
6. $\to r$ — apply continued fractions algorithm

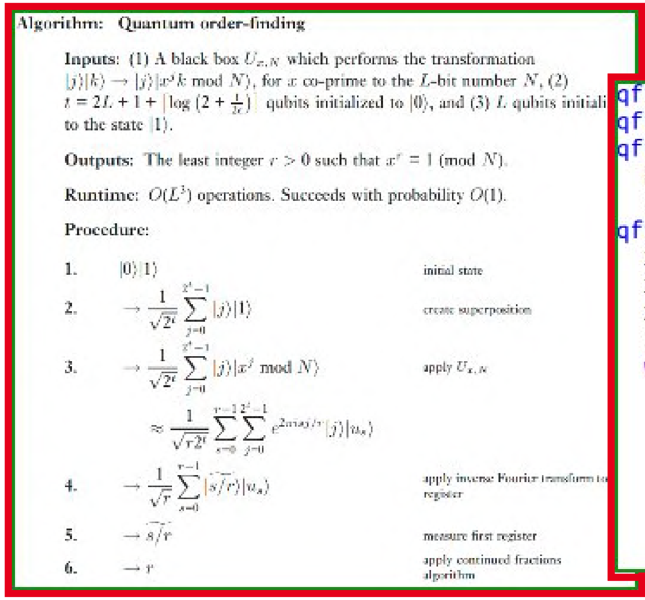
```
qft_internal :: [Qubit] -> Circ [Qubit]
qft_internal [] = return []
qft_internal [x] = do
  hadamard x
  return [x]
qft_internal (x:xs) = do
  xs' <- qft_internal xs
  xs'' <- rotations x xs' (length xs')
  x' <- hadamard x
  return (x':xs'')
  where
    -- Auxiliary function used by 'qft'.
    rotations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



A specification preamble:

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

Verification : specifications


Algorithm: Quantum order-finding

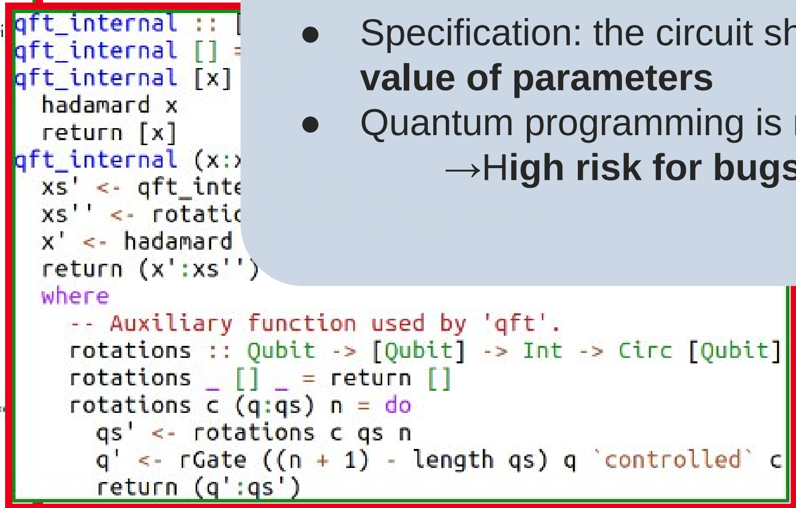
Inputs: (1) A black box $U_{x,N}$ which performs the transformation $|j\rangle|k\rangle \rightarrow |j\rangle|x^j k \bmod N\rangle$, for x co-prime to the L-bit number N, (2) $t = 2L + 1 + \lceil \log(2 + \frac{1}{2\epsilon}) \rceil$ qubits initialized to $|0\rangle$, and (3) L qubits initiali[zed] to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

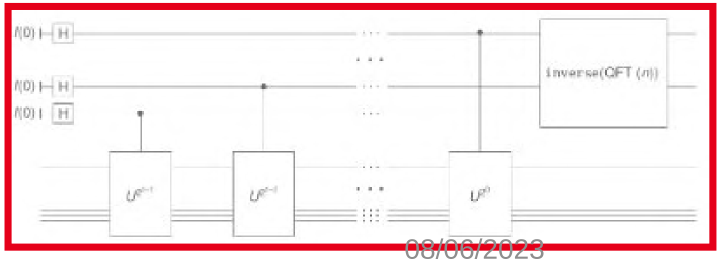
1. $|0\rangle|1\rangle$ — initial state
2. $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|1\rangle$ — create superposition
3. $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|x^j \bmod N\rangle$ — apply $U_{x,N}$
   $\approx \frac{1}{\sqrt{r2^t}} \sum_{s=0}^{r-1} \sum_{j=0}^{2^t-1}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


```
qft_internal :: [
qft_internal [] =
qft_internal [x]
  hadamard x
  return [x]
qft_internal (x:x
  xs' <- qft_inte
  xs'' <- rotatio
  x' <- hadamard
  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')
```

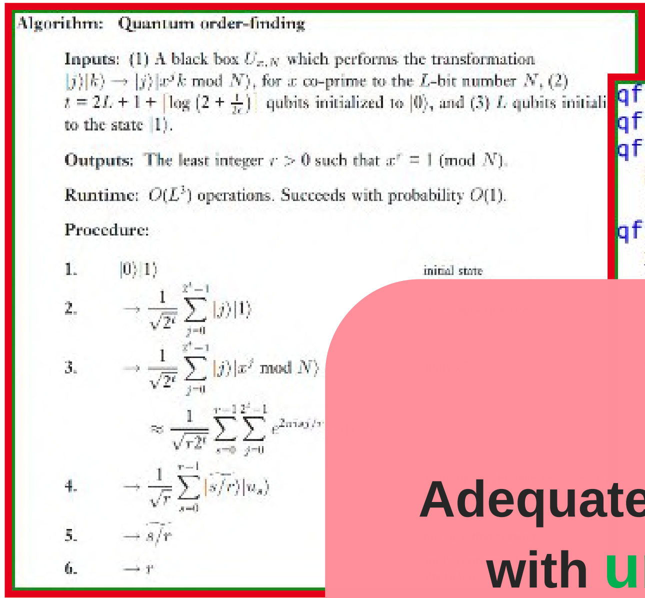
- Specification: the circuit should meet the spec **for any value of parameters**
- Quantum programming is non-intuitive
  →**High risk for bugs**!

A specification preamble:

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



Verification : specifications


Algorithm: Quantum order-finding

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

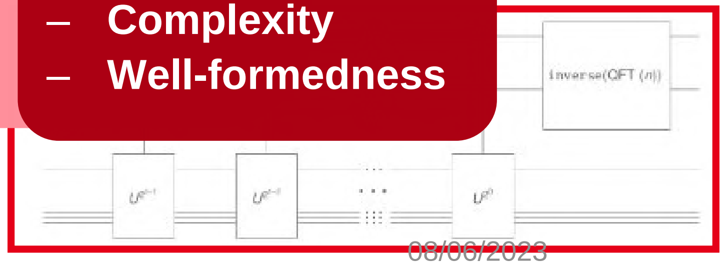
```
qft_internal :: [
qft_internal [] =
qft_internal [x]
  hadamard x
  return [x]
qft_internal (x:x
  xs' <- qft_inte
```

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

A specification [...]
- **Input parame**[...]
- **Functional co**[...]
- **Complexity**: number of elementary operations

Adequate implementation should come with universally valid evidence regarding the specs

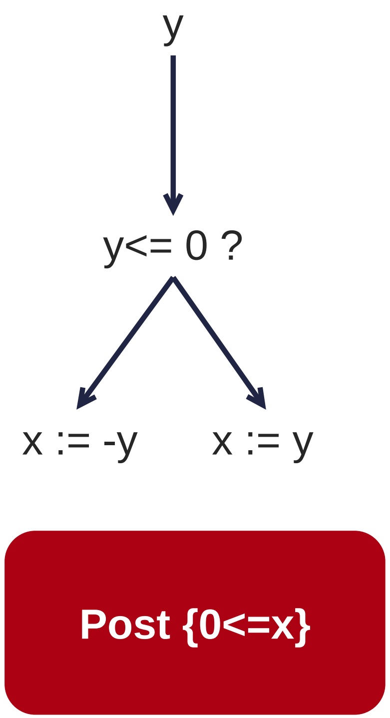
- **Functionality**
- **Complexity**
- **Well-formedness**



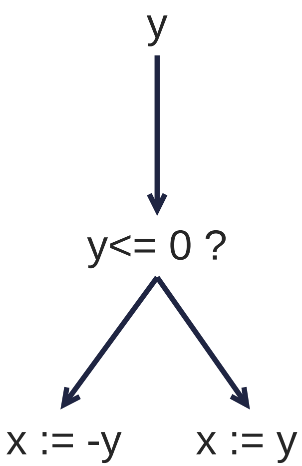
Standard debuguing techniques fail…

Potential method	Drawback
Assertion checking ?	Requires (destructive) **measurement** with highly superposed states
Final test ?	How to pinpoint **error source** ?
Simulation ?	As far as we don't need a Quantum Computer !

Static analysis ? A naive example...



Static analysis ? A naive example...



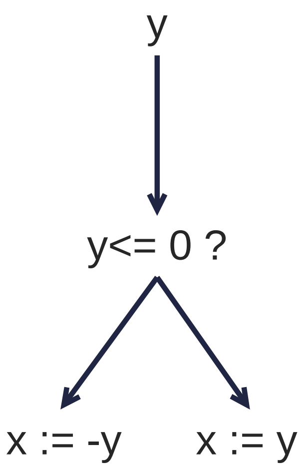
Test :
- Case y =0
- Case y =-3
- Case y =27

.....

-What about y = 17 ?
-What about y = 128... ...123 ?

Post {0<=x}

Static analysis ? A naive example…



Post {0<=x}

Test :

- Case y =0
- Case y =-3
- Case y =27

…..

-What about y = 17 ?
-What about y = 128… …123 ?

Static reasoning :

$$\frac{y \leq 0 \mid y > 0 \qquad \begin{array}{c} y <= 0 \\ \ldots \\ 0 <= x \end{array} \qquad \begin{array}{c} y > 0 \\ \ldots \\ 0 <= x \end{array}}{0 <= x}$$

Standard debuguing techniques fail...
... the alternative of formal verification

Potential method	Drawback
Assertion checking ?	Requires (destructive) **measurement** with highly superposed states
Final test ?	How to pinpoint **error source** ?
Simulation ?	As far as we don't need a Quantum Computer !

Testing/Assertion checking	Formal verification
executions/simulations	**static** analysis, **no need to execute**
bounded parameters	**scale insensitive/any instance**
statistical arguments	absolute, **mathematical guarantee**

Build on **best practice** of formal verification for the classical case and **tailor them to the quantum** case

QBRICKS

Standard debuguing techniques fail…
… the alternative of formal verification

Co-design

Needs **3 main ingredients :**

- Formal semantics
- Spec language
- Proof engine
  + object language

Testing/Assertion checking	Formal verification
executions/simulations	**static** analysis, **no need to execute**
b**ounded** parameters	**scale insensitive/any instance**
statistical arguments	absolute, **mathematical guarantee**

Build on **best practice** of formal verification for the classical case and **tailor them to the quantum** case

Quantum programming and formal verification

1. Quantum programming, program specifications and verification

2. **Main challenges**

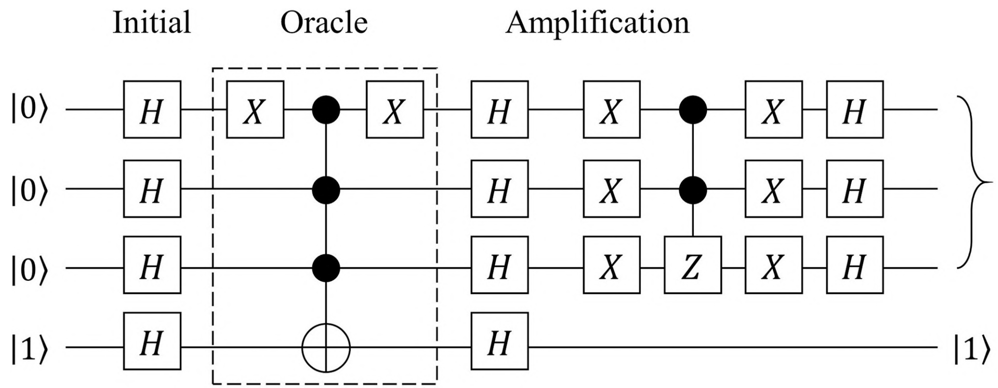
 - **co-design object/specifications languages**

 - **symbolic representation for standard programing features**

3. A word about our works

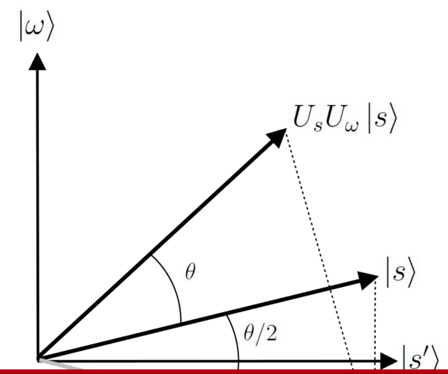
User-friendly programming languages

The current quantum programming solutions rely on **sequential descriptions** of elementary quantum operations, similar to classical **assembly programs**.



- How to **hold the « big picture »** ?
- Unavoidable **side reasoning** in a « formal » setting
  - formal interpretation language on top of the object programming language ?

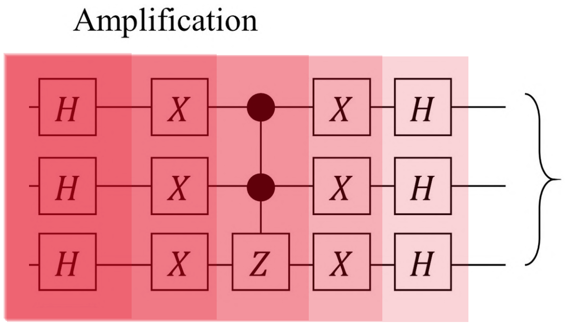
- The need for **programming** features/primitives ...
  - **High-levelled**, as far as possible
  - With **intuitive** procedural meaning and/but ...
  - ... Formally **interpretable**
- **...** and for characteizing this « formally »



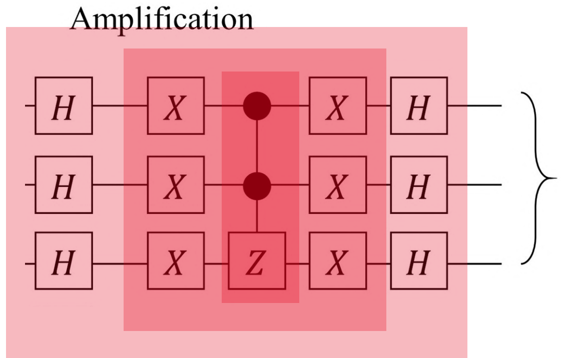
Formal reasoning is natural !

Co-designing object language/proof engine

– Path-sums for Grover diffusor :



$$|x\rangle_{comp} \rightarrow \frac{1}{\sqrt{2^{2n}}}\sum_{y \in BV_{2n}} e^{i\pi\left(\frac{\vec{x}\vec{y_1}}{2}+\frac{\vec{y_1}\vec{y_2}}{2}+\frac{\prod \overline{y_1^i}}{2}\right)} |k(x,y)\rangle$$



$$|x\rangle_{HX^{\otimes n}} \rightarrow e^{i\pi\left(\frac{\prod x}{2}\right)} |x\rangle$$

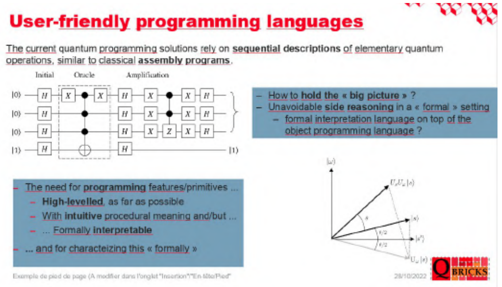
```
| diffusor | (qreg qr, qreg aux)
  circ qr, aux ->
    with conjugated (H(qr)) {
      with conjugated (X(qr)) {
```

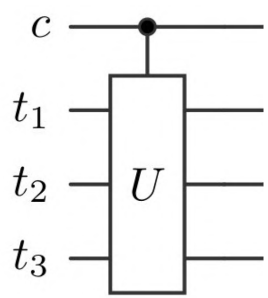
User friendly programming features
//
Tractable formal representation

– Interface :
  trade-off **established view ⇔ formal reasoning forecast**
– Further extensions : path-sums splitting, linear combinations of PS,etc

Symbolic representations, the case of subcircuit control

- Modular reasoning only for sequence/parallelism → **assembly code**
- Any **higher-level** consideration requires **adaptation** : eg. Subcircuit control

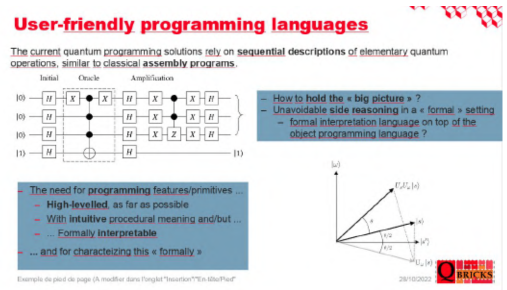




State			Density operator		
$\vert\bullet\rangle_1$	=	$\alpha\vert 0\rangle_1 + \beta\vert 1\rangle_1$	$\vert x\rangle\langle x\vert$	=	$\begin{pmatrix} \alpha\overline{\alpha} & \alpha\overline{\beta} \\ \beta\overline{\alpha} & \beta\overline{\beta} \end{pmatrix}$
$e^{i\pi\theta}\vert\bullet\rangle_1$	=	$e^{i\pi\theta}\alpha\vert 0\rangle_1 + e^{i\pi\theta}\beta\vert 1\rangle_1$	$e^{i\pi\theta}\vert x\rangle\langle x\vert e^{-i\pi\theta}$	=	$\begin{pmatrix} \alpha\overline{\alpha} & \alpha\overline{\beta} \\ \beta\overline{\alpha} & \beta\overline{\beta} \end{pmatrix}$
$\vert\bullet\rangle_0?(\vert\bullet\rangle_1; e^{i\pi\theta}\vert\bullet\rangle_1)$	=	$\alpha_0\alpha_1\vert 0\rangle_0\vert 0\rangle_1 +$ $\alpha_0\beta_1\vert 0\rangle_0\vert 1\rangle_1 +$ $e^{i\pi\theta}\beta_0\alpha_1\vert 1\rangle_0\vert 0\rangle_1 +$ $e^{i\pi\theta}\beta_0\beta_1\vert 1\rangle_0\vert 1\rangle_1$	$(\vert 0\rangle\langle 0\vert\vert x\rangle_1 + e^{-i\pi\theta}\vert 1\rangle\langle 1\vert\vert x\rangle_1)$ $(e^{i\pi\theta}\ \langle x\vert_1\vert 1\rangle\langle 1\vert + \langle x\vert_1\vert 0\rangle\langle 0\vert)$	=	$\begin{pmatrix} \alpha_0\alpha_1\overline{\alpha_0\alpha_1} & \alpha_0\alpha_1\overline{\alpha_0\beta_1} & e^{-i\pi\theta}\alpha_0\alpha_1\overline{\beta_0\alpha_1} & e^{-i\pi\theta}\alpha_0\alpha_1\overline{\beta_0\beta_1} \\ \alpha_0\beta_1\overline{\alpha_0\alpha_1} & \alpha_0\beta_1\overline{\alpha_0\beta_1} & e^{-i\pi\theta}\alpha_0\beta_1\overline{\beta_0\alpha_1} & e^{-i\pi\theta}\alpha_0\beta_1\overline{\beta_0\beta_1} \\ e^{i\pi\theta}\beta_0\alpha_1\overline{\alpha_0\alpha_1} & e^{i\pi\theta}\beta_0\alpha_1\overline{\alpha_0\beta_1} & \beta_0\alpha_1\overline{\beta_0\alpha_1} & \beta_0\alpha_1\overline{\beta_0\beta_1} \\ e^{i\pi\theta}\beta_0\beta_1\overline{\alpha_0\alpha_1} & e^{i\pi\theta}\beta_0\beta_1\overline{\alpha_0\beta_1} & \beta_0\beta_1\overline{\beta_0\alpha_1} & \beta_0\beta_1\overline{\beta_0\beta_1} \end{pmatrix}$
$\vert\bullet\rangle_0?(\vert\bullet\rangle_1; \vert\bullet\rangle_1))$	=	$\alpha_0\alpha_1\vert 0\rangle_0\vert 0\rangle_1 +$ $\alpha_0\beta_1\vert 0\rangle_0\vert 1\rangle_1 +$ $\beta_0\alpha_1\vert 1\rangle_0\vert 0\rangle_1 +$ $\beta_0\beta_1\vert 1\rangle_0\vert 1\rangle_1$	$(\vert 0\rangle\langle 0\vert\vert x\rangle_1 + \vert 1\rangle\langle 1\vert\vert x\rangle_1)$ $(\ \langle x\vert_1\vert 1\rangle\langle 1\vert + \langle x\vert_1\vert 0\rangle\langle 0\vert)$	=	$\begin{pmatrix} \alpha_0\alpha_1\overline{\alpha_0\alpha_1} & \alpha_0\alpha_1\overline{\alpha_0\beta_1} & \alpha_0\alpha_1\overline{\beta_0\alpha_1} & \alpha_0\alpha_1\overline{\beta_0\beta_1} \\ \alpha_0\beta_1\overline{\alpha_0\alpha_1} & \alpha_0\beta_1\overline{\alpha_0\beta_1} & \alpha_0\beta_1\overline{\beta_0\alpha_1} & \alpha_0\beta_1\overline{\beta_0\beta_1} \\ \beta_0\alpha_1\overline{\alpha_0\alpha_1} & \beta_0\alpha_1\overline{\alpha_0\beta_1} & \beta_0\alpha_1\overline{\beta_0\alpha_1} & \beta_0\alpha_1\overline{\beta_0\beta_1} \\ \beta_0\beta_1\overline{\alpha_0\alpha_1} & \beta_0\beta_1\overline{\alpha_0\beta_1} & \beta_0\beta_1\overline{\beta_0\alpha_1} & \beta_0\beta_1\overline{\beta_0\beta_1} \end{pmatrix}$

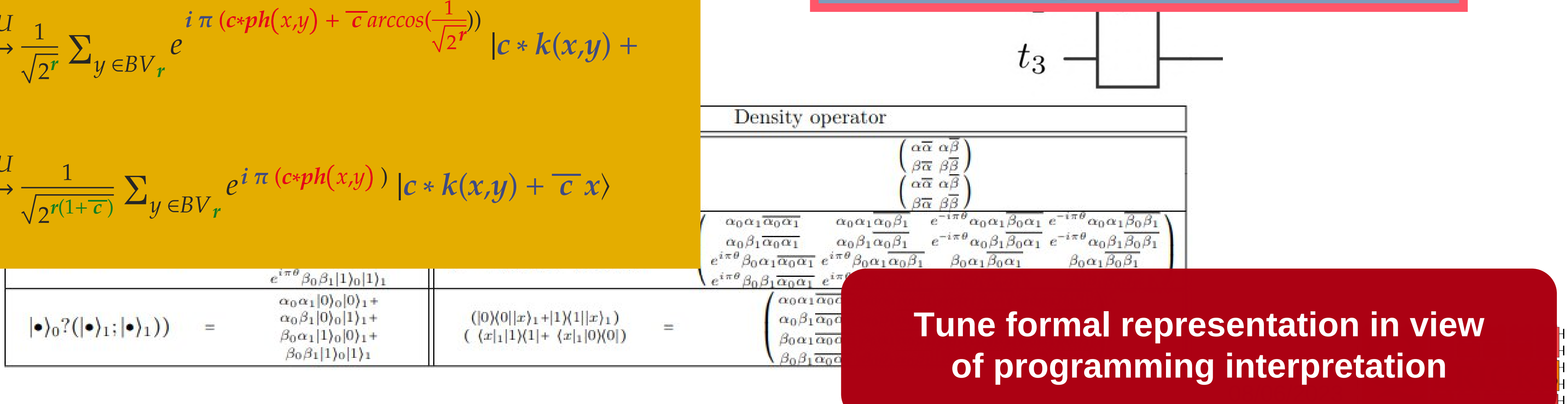
Symbolic representations, the case of subcircuit control

- Modular reasoning only for sequence/parallelism → **assembly code**
- Any **higher-level** consideration requires **adaptation** : eg. Subcircuit control



- $|x\rangle \xrightarrow{U} \frac{1}{\sqrt{2^r}} \sum_{y \in BV_r} e^{i\pi\, ph(x,y)} |k(x,y)\rangle$
- $|x\rangle \xrightarrow{C-U} \frac{1}{\sqrt{2^r}} \sum_{y \in BV_r} e^{i\pi\,(c*ph(x,y) + \overline{c}\arccos(\frac{1}{\sqrt{2^r}}))} |c*k(x,y) + \overline{c}\, x\rangle$
- $|x\rangle \xrightarrow{C-U} \frac{1}{\sqrt{2^{r(1+\overline{c})}}} \sum_{y \in BV_r} e^{i\pi\,(c*ph(x,y))} |c*k(x,y) + \overline{c}\, x\rangle$

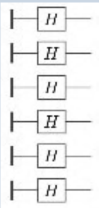
Trade-off SR **computing performance** Vs **expressivity**

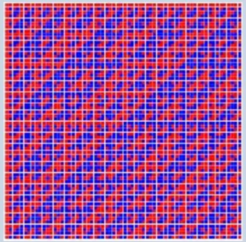


Tune formal representation in view of programming interpretation

Symbolic representations, the case of measurement

Standard interpretation as matrices :





- Cumbersome
- Requires **higher-order reasoning**

$$x \quad := \quad |x\rangle\langle x|$$

$$|x\rangle \xrightarrow{A} \mathbf{MAT}(A)\cdot|x\rangle\langle x|\cdot\mathbf{MAT}(A)^\dagger$$

$$= |\mathbf{MAT}(A)\cdot x\rangle\langle\mathbf{MAT}(A)\cdot x|^\dagger$$

$$|x\rangle \xrightarrow{(A--B)} \mathbf{MAT}(A--B)\cdot|x\rangle\langle x|\cdot\mathbf{MAT}(A--B)^\dagger$$

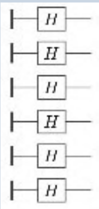
$$= |\mathbf{MAT}(B)\cdot\mathbf{MAT}(A)\cdot x\rangle\langle\mathbf{MAT}(B)\cdot\mathbf{MAT}(A)\cdot x|^\dagger$$

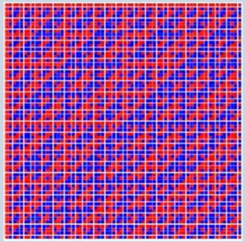
$$|x\rangle \xrightarrow{Meas_m} \frac{\mathbf{Meas}_m\cdot|x\rangle\langle x|\cdot\mathbf{Meas}_m^\dagger}{tr(\mathbf{Meas}_m^\dagger\mathbf{Meas}_m|x\rangle\langle x|)}$$

$$\mathbf{Meas}_m = \sum_j \left(\mathbf{MAT}(ID)^{\otimes i}\otimes|j\rangle\langle j|^\dagger\otimes\mathbf{MAT}(ID)^{\otimes k}\right)$$

Symbolic representations, the case of measurement

Standard interpretation as matrices :





- Cumbersome
- Requires **higher-order reasoning**

$$x \quad := \quad |x\rangle\langle x|$$

$$|x\rangle \xrightarrow{A} \mathbf{MAT}(A)\cdot|x\rangle\langle x|\cdot\mathbf{MAT}(A)^{\dagger}$$

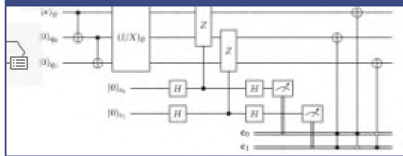
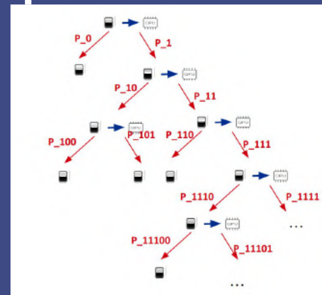
$$= |\mathbf{MAT}(A)\cdot x\rangle\langle\mathbf{MAT}(A)\cdot x|^{\dagger}$$

$$|x\rangle \xrightarrow{(A--B)} \mathbf{MAT}(A--B)\cdot|x\rangle\langle x|\cdot\mathbf{MAT}(A--B)^{\dagger}$$

$$|x\rangle$$

Open problem : to find **unified tractable** symbolic **representations**

$$\mathbf{Meas}_m = \sum_j \left(\mathbf{MAT}(ID)^{\otimes i}\otimes|j\rangle\langle j|^{\dagger}\otimes\mathbf{MAT}(ID)^{\otimes k}\right)$$

Low-level Processes	High-level processes
	
Continuous	Discrete
Deterministic	Probabilistic
Unitary	Non-unitary

	L	H	Meas.	C. V.	Q. C.	Par.	Impl	Aut.	SR
SQIR[24]	⇌								DO
CoqQ[53]									DO
PyZx[27]									ZX
QHL[33]									DO
QHL[18]									DO
SOP[1]									SOP
SOP[46]									SOP
OBRICKS[10]									SOP

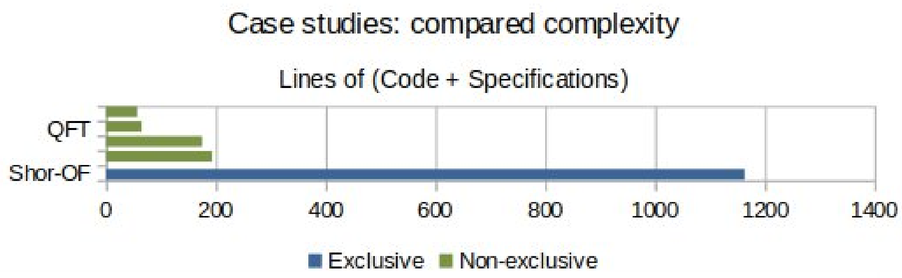
Manuscript submitted to ACM

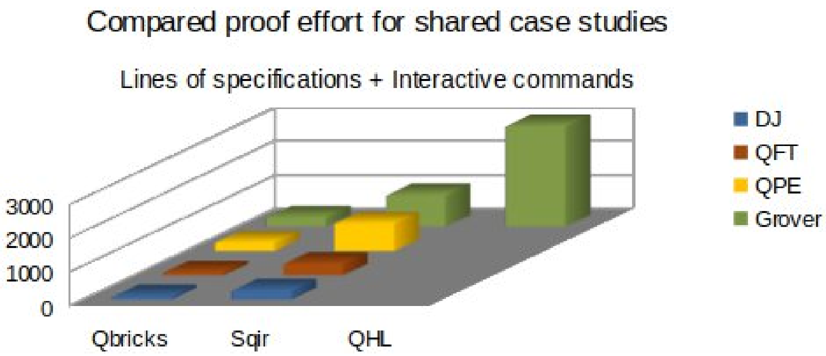
Quantum programming and formal verification

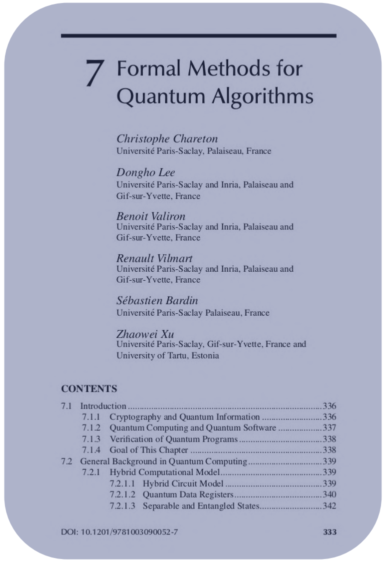
Qbricks core : achievements

MAJOR ACHIEVEMENTS

- a core development framework for **parametrized verified quantum programming**
- **first ever verified implementation of Shor order finding algorithm** (95% proof automation),







7 Formal Methods for Quantum Algorithms

Christophe Chareton
Université Paris-Saclay, Palaiseau, France

Dongho Lee
Université Paris-Saclay and Inria, Palaiseau and Gif-sur-Yvette, France

Benoit Valiron
Université Paris-Saclay and Inria, Palaiseau and Gif-sur-Yvette, France

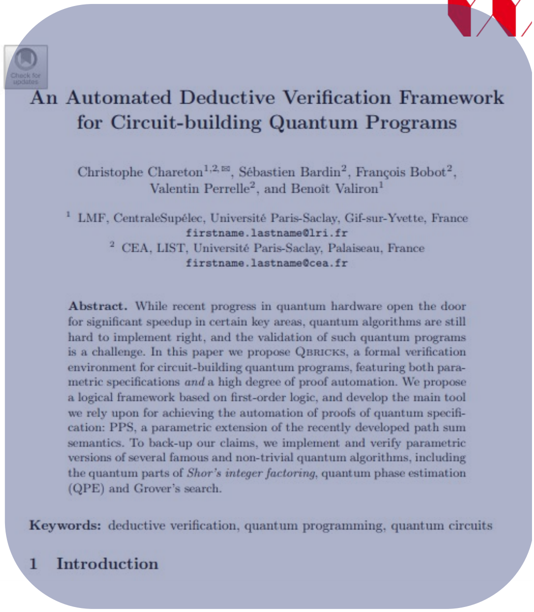
Renault Vilmart
Université Paris-Saclay and Inria, Palaiseau and Gif-sur-Yvette, France

Sébastien Bardin
Université Paris-Saclay Palaiseau, France

Zhaowei Xu
Université Paris-Saclay, Gif-sur-Yvette, France and University of Tartu, Estonia

CONTENTS

DOI: 10.1201/9781003090052-7 333



Check for updates

An Automated Deductive Verification Framework for Circuit-building Quantum Programs

Christophe Chareton[1,2,✉], Sébastien Bardin[2], François Bobot[2], Valentin Perrelle[2], and Benoît Valiron[1]

[1] LMF, CentraleSupélec, Université Paris-Saclay, Gif-sur-Yvette, France
firstname.lastname@lri.fr
[2] CEA, LIST, Université Paris-Saclay, Palaiseau, France
firstname.lastname@cea.fr

Abstract. While recent progress in quantum hardware open the door for significant speedup in certain key areas, quantum algorithms are still hard to implement right, and the validation of such quantum programs is a challenge. In this paper we propose QBRICKS, a formal verification environment for circuit-building quantum programs, featuring both parametric specifications *and* a high degree of proof automation. We propose a logical framework based on first-order logic, and develop the main tool we rely upon for achieving the automation of proofs of quantum specification: PPS, a parametric extension of the recently developed path sum semantics. To back-up our claims, we implement and verify parametric versions of several famous and non-trivial quantum algorithms, including the quantum parts of *Shor's integer factoring*, quantum phase estimation (QPE) and Grover's search.

Keywords: deductive verification, quantum programming, quantum circuits

1 Introduction

QBRICKS

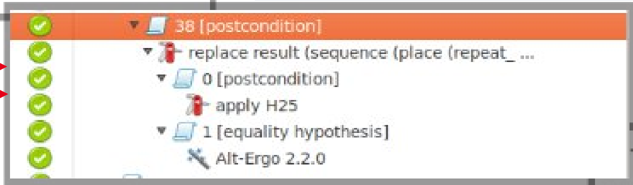
Toward a formally verified stack : first prototype

Imbricks code for qft(k)

- 7 lines of codes
- 13 lines of specifications
  - Functional specs : $|j\rangle \longrightarrow \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{2\pi ijk/N}|k\rangle$
  - Performance specs : Size ≤ c•k^2
  - Well-formedness

```
|| qft || (qreg qr)
      circ qr ->
      for q in range(len(qr)) {
         H(qr[q])
         for i in range(qr[q+1..-1]) {
            with control qr[i+1] (RZ(i-q, qr[q]))
      return
```

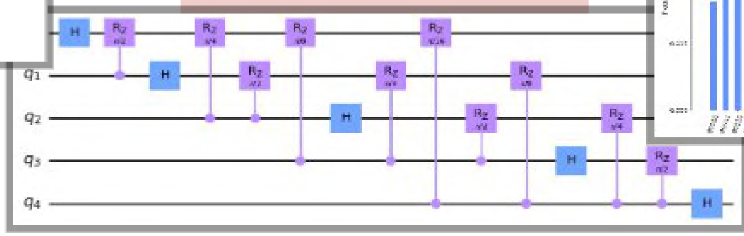
12 interactive commands to guide the proof



Mathematical theorems library

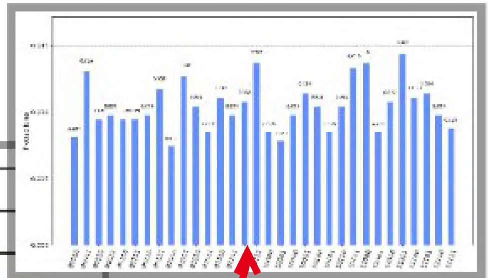
Standard Oqasm IR for instances of k :

k	Lines of code
5	20
20	200
80	3200



IBM simulator

k	Simulation time
5	4’’
20	10’’
80	Non feasible



Imbricks → QBRICKS → **Oqasm** → **Simulation**

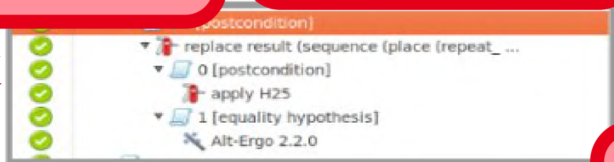
Toward a formally verified stack : first prototype

Imbricks code for qft(k)
- 7 lines of codes
- 13 lines of specifications
  - Functional specs
  - Performance spe
  - Well-formedness

Hybrid SR + reasoning

Formal reasoning driven syntax

Proof acceleration



Mathematical theorems library

- **Integration in the national strategy**
  - **PEPR EPIC:** (1 task lead/ collabs ac LMF,DSCIN,DILS ...)
  - **Initiative HQI:** (1 WP lead)

Standard Oqasm IR for instances of k :

k	Lines of code
5	20
20	200
80	3200

	time
5	4’’
20	10’’
80	Non feasible

Proven compilation

Imbricks → QBRICKS → **Oqasm** → **Simulation**

Any question ?

Christophe Chareton
Christophe.chareton@cea.fr