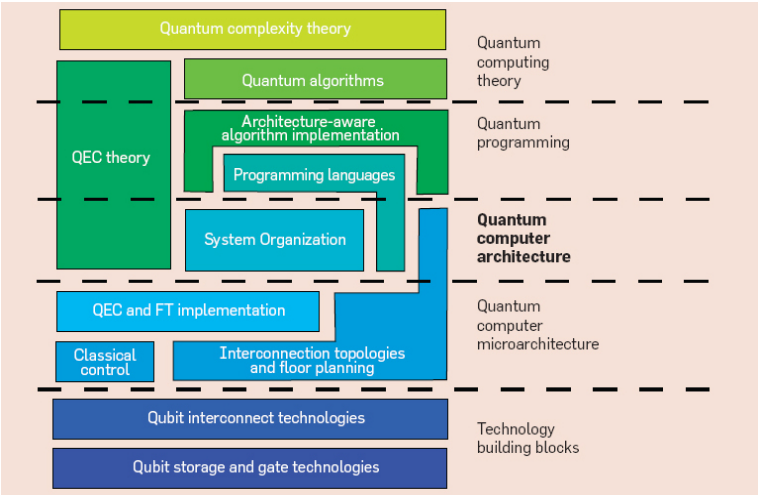
universität innsbruck

Formal Methods for Quantum Programs

Martin Avanzini **Georg Moser** Romain Péchoux Simon Perdrix
Vladimir Zamdzhiev

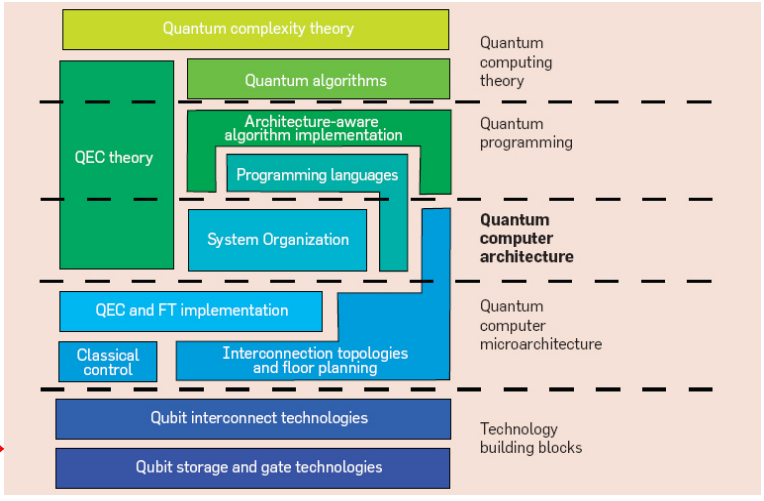
https://tcs-informatik.uibk.ac.at

A Bit of Context



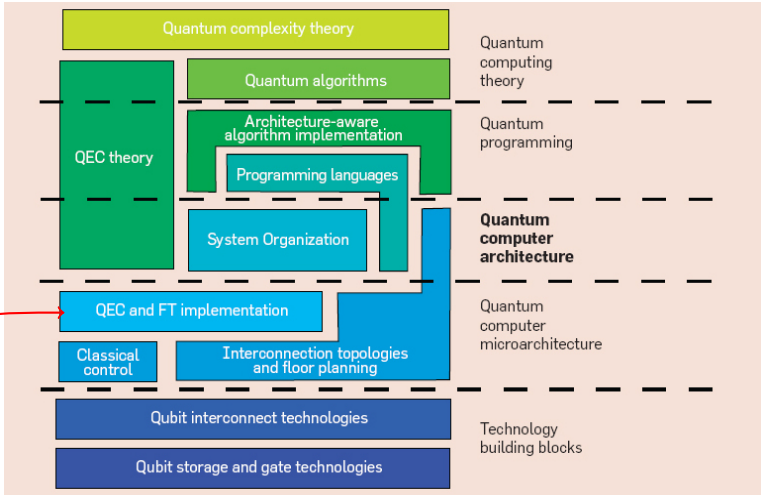
R. V. Meter and C. Horsman.
A blueprint for building a quantum computer. *Commun. ACM*, 56(10):84–93, 2013.

A Bit of Context



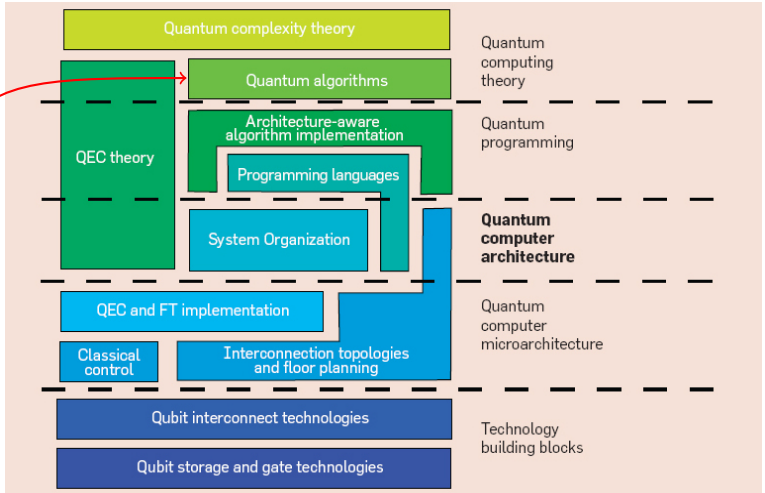
H. Briegel, D. Browne, W. Dür, R. Raussendorf, and M. V. den Nest. Measurement-based quantum computation. *Nature Physics*, 5:19–26, 2009.

A Bit of Context



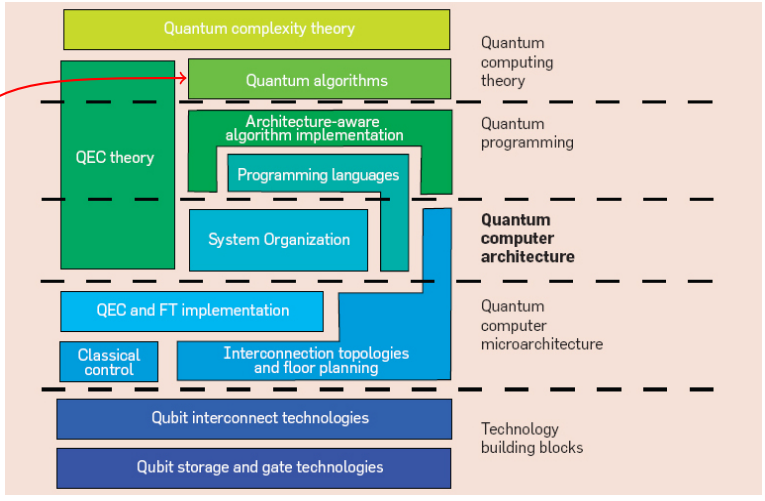
P. Schindler, J. Barreiro, T. Monz, V. Nebendahl, D. Nigg, M. Chwalla, M. Hennrich, and R. Blatt. Experimental repetitive quantum error correction. *Science*, 332(6033), 2011.

A Bit of Context



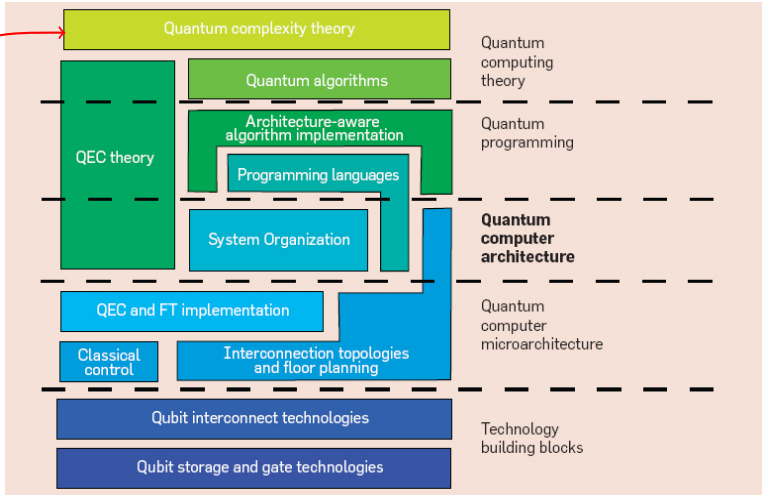
V. Dunjko and H. Briegel.
Machine learning & artificial intelligence in the quantum domain. *arXiv*, (1709.02779), 2017.

A Bit of Context



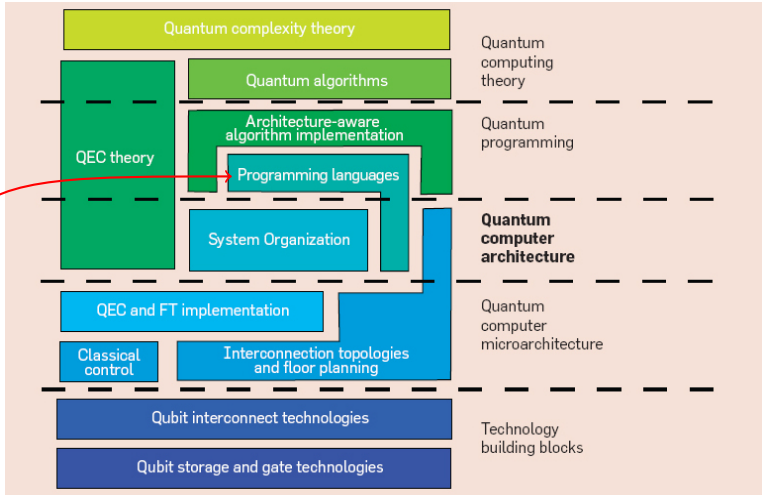
H. Miyahara, K. Aihara, and W. Lechner.
Quantum expectation-maximization algorithm. *Physical Review A*, 101(1), 2020.

A Bit of Context



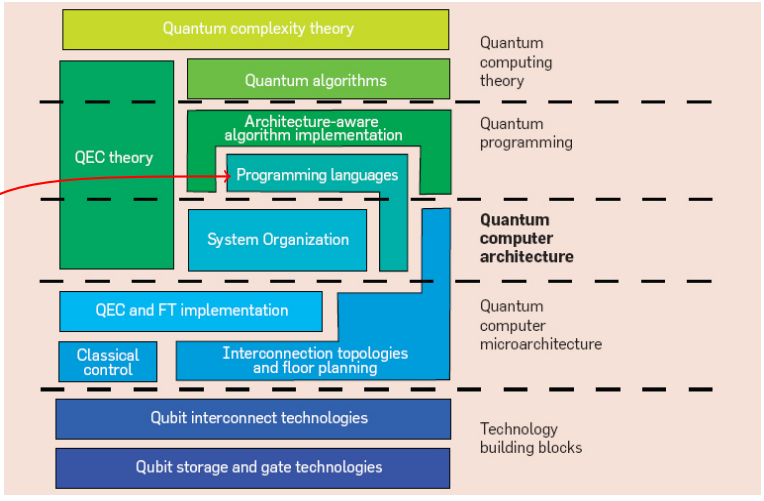
V. Torggler, P. Aumann, H. Ritsch, and W. Lechner.
A quantum n-queens solver. *Quantum*, 3(149), 2019.

A Bit of Context



V. Danos, E. Kashefi, and P. Panangaden.
The measurement calculus. *J. ACM*, 54(2):8, 2007.

A Bit of Context



P. Selinger.
Towards a quantum programming language. *Math. Struct. Comput. Sci.*, 14(4):527–586, 2004.

Motivation to Study Quantum Computation

Viewpoint of Computer Science

1. quantum effects allows us to break the “strong” form of the Church-Turing thesis—any reasonable model of computation is efficiently computable by a Turing machine

Motivation to Study Quantum Computation

Viewpoint of Computer Science

1. quantum effects allows us to break the “strong” form of the Church-Turing thesis—any reasonable model of computation is efficiently computable by a Turing machine
2. theoretical computer science is about the study of computation, regardless of the physical device implementing this computation

Motivation to Study Quantum Computation

Viewpoint of Computer Science

1. quantum effects allows us to break the “strong” form of the Church-Turing thesis—any reasonable model of computation is efficiently computable by a Turing machine
2. theoretical computer science is about the study of computation, regardless of the physical device implementing this computation
3. since our understanding of the world is quantum mechanical, TCS has to study quantum computers, not classical ones

Motivation to Study Quantum Computation

Viewpoint of Computer Science

1. quantum effects allows us to break the “strong” form of the Church-Turing thesis—any reasonable model of computation is efficiently computable by a Turing machine
2. theoretical computer science is about the study of computation, regardless of the physical device implementing this computation
3. since our understanding of the world is quantum mechanical, TCS has to study quantum computers, not classical ones
4. more practically, much of cryptography becomes insecure in a quantum world, while entanglement makes it possible to design unconditionally secure key distribution: post-quantum cryptography

Motivation to Study Quantum Computation

Viewpoint of Computer Science

1. quantum effects allows us to break the “strong” form of the Church-Turing thesis—any reasonable model of computation is efficiently computable by a Turing machine
2. theoretical computer science is about the study of computation, regardless of the physical device implementing this computation
3. since our understanding of the world is quantum mechanical, TCS has to study quantum computers, not classical ones
4. more practically, much of cryptography becomes insecure in a quantum world, while entanglement makes it possible to design unconditionally secure key distribution: post-quantum cryptography
5. similarly, as quantum computers become the new normal, a theory of quantum programming languages is crucially needed

- **Quantum Programming Languages**
- **Program and Resource Analysis**
- **Expected Cost Analysis for Quantum Programs**

universität innsbruck

Quantum Programming Languages

Requirements for Quantum Programming Languages

Allocation and Measurement

- allocate and measure quantum registers
- apply unitary operations

Requirements for Quantum Programming Languages

Allocation and Measurement

- allocate and measure quantum registers
- apply unitary operations

Reasoning about Subroutines

- defining and calling subroutines
- reversing subroutines

Requirements for Quantum Programming Languages

Allocation and Measurement

- allocate and measure quantum registers
- apply unitary operations

Reasoning about Subroutines

- defining and calling subroutines
- reversing subroutines

Building Quantum Oracles

- build quantum oracles from classical functions

Quantum Data Types

- data types allow to reason abstractly
- (quantum) data types allow for static type checking thus providing compile-time guarantees

Quantum Data Types

- data types allow to reason abstractly
- (quantum) data types allow for static type checking thus providing compile-time guarantees

Specification and Verification

- formal specification and verification
- operational or denotational semantics

Quantum Data Types

- data types allow to reason abstractly
- (quantum) data types allow for static type checking thus providing compile-time guarantees

Specification and Verification

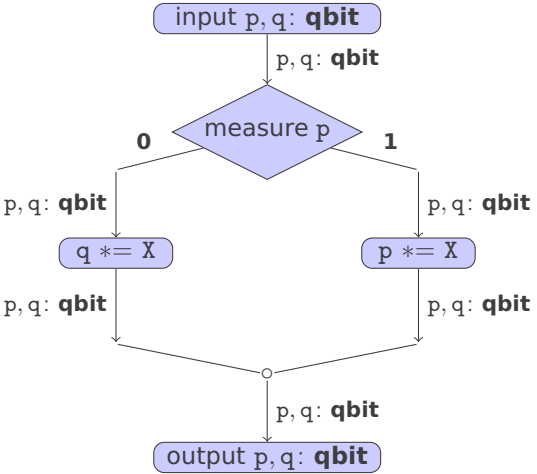
- formal specification and verification
- operational or denotational semantics

Resource Sensitivity and Resource Estimation

- estimation of resource requirements (eg. number of elementary gates, expected runtime)
- transparancies of quantum error correction

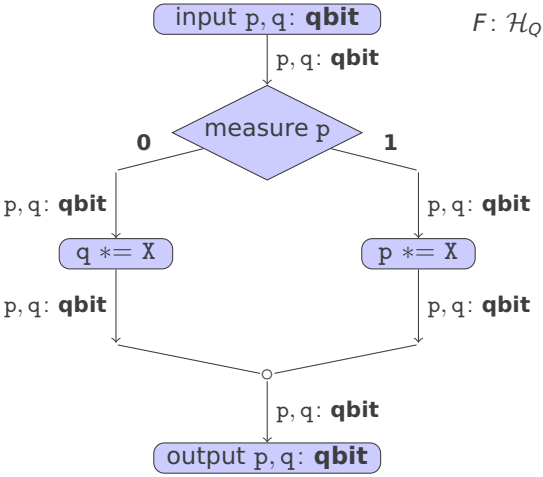
Selinger's Quantum Programming Language

Quantum Flow Language

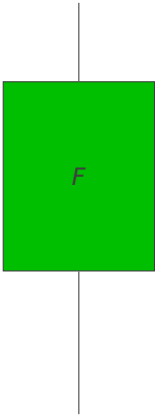


Selinger's Quantum Programming Language

Quantum Flow Language

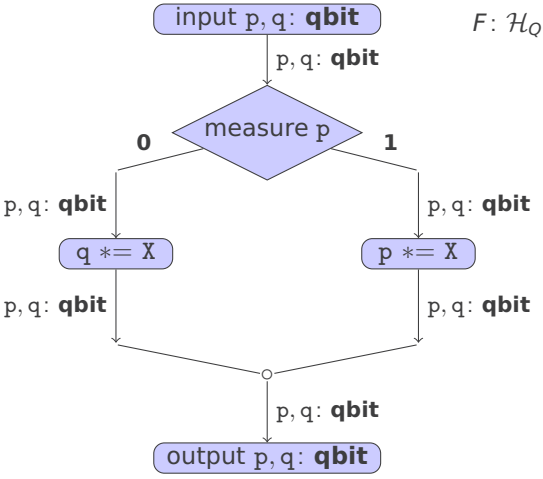


$F\colon \mathcal{H}_Q \xrightarrow{?} \mathcal{H}_Q$

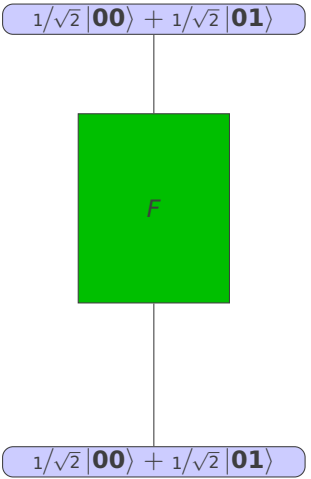


Selinger's Quantum Programming Language

Quantum Flow Language

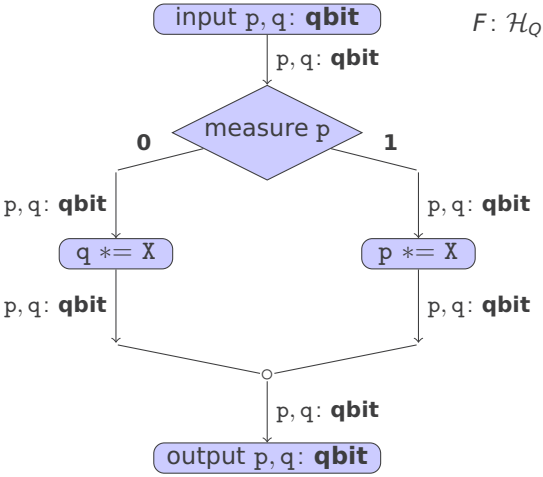


$F\colon \mathcal{H}_Q \xrightarrow{?} \mathcal{H}_Q$

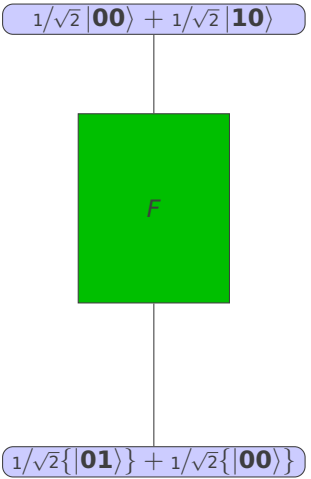


Selinger's Quantum Programming Language

Quantum Flow Language

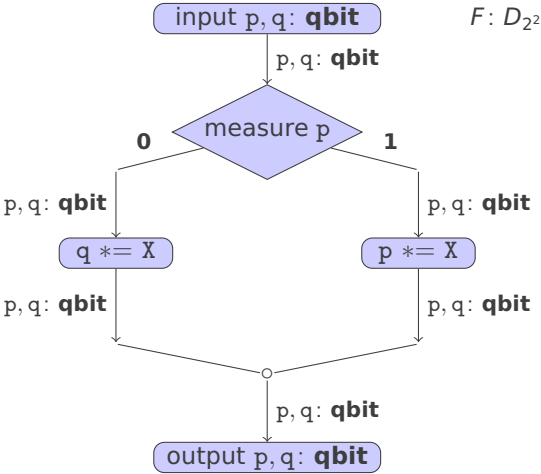


$F\colon \mathcal{H}_Q \xrightarrow{?} \mathcal{H}_Q$

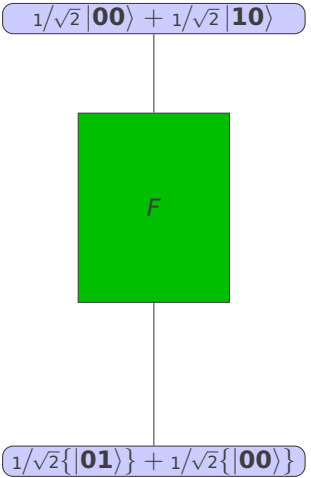


Selinger's Quantum Programming Language

Quantum Flow Language

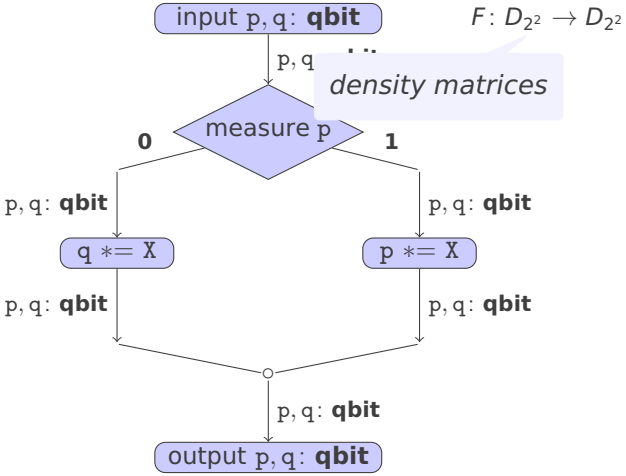


$F : D_{2^2} \to D_{2^2}$

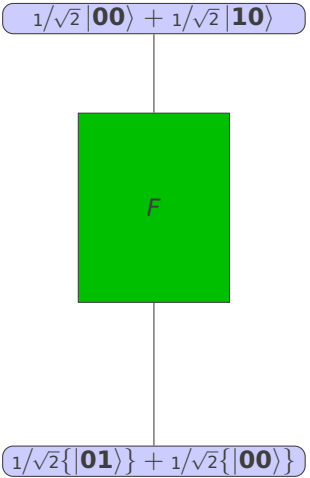


Selinger's Quantum Programming Language

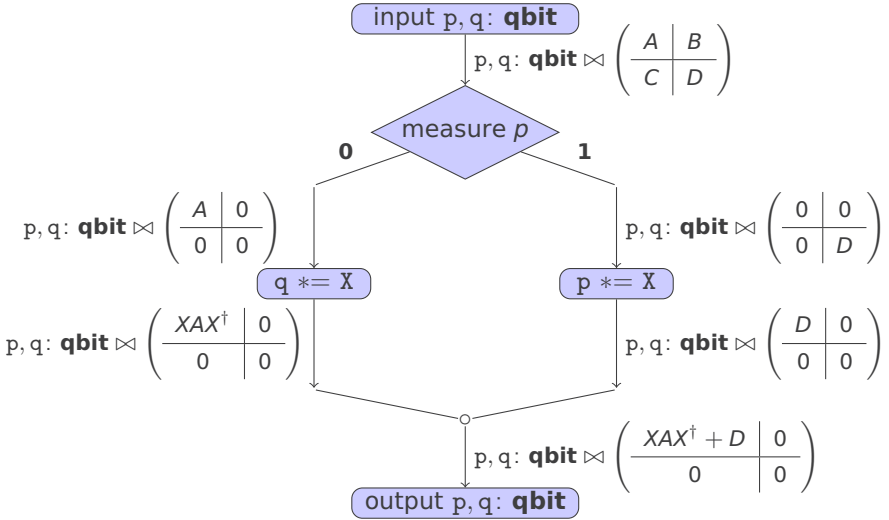
Quantum Flow Language



$F : D_{2^2} \to D_{2^2}$

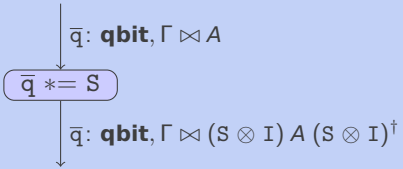


Quantum Flows Language (cont'd)



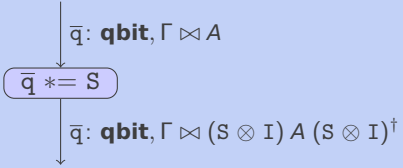
Focus on Unitary Transformation, Merge and Measurement

Unitary Transformation



Focus on Unitary Transformation, Merge and Measurement

Unitary Transformation

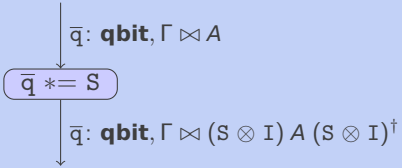


Example (Elementary Gates)

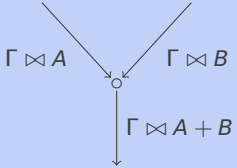
$$X := \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad H := \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad T := \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix}$$

Focus on Unitary Transformation, Merge and Measurement

Unitary Transformation



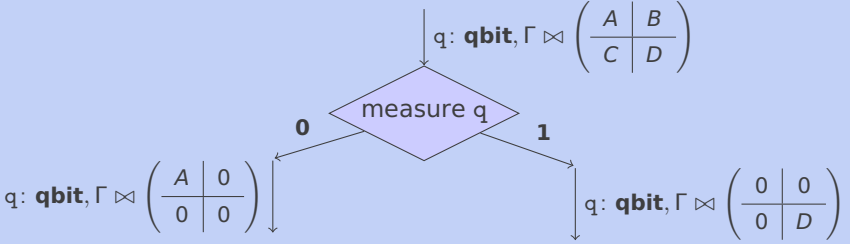
Merge



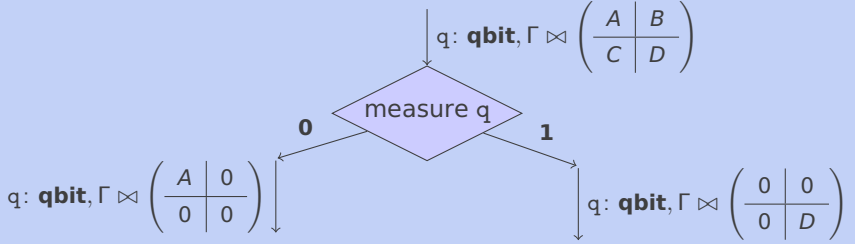
Example (Elementary Gates)

$$X := \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad H := \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad T := \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix}$$

Measurement



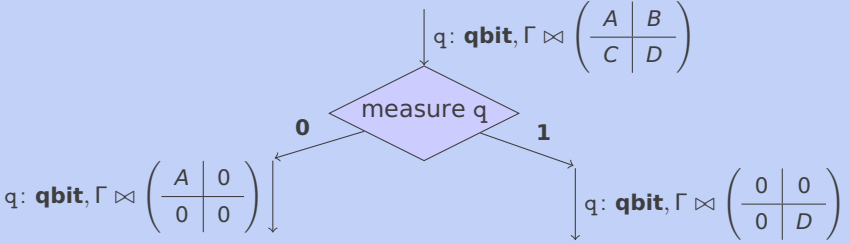
Measurement



- let $|\varphi\rangle$ be a pure state, s.t.

$$|\varphi\rangle = \begin{pmatrix} v \\ \hline w \end{pmatrix} \qquad |\varphi\rangle\langle\varphi| = \left(\begin{array}{c|c} vv^\dagger & vw^\dagger \\ \hline wv^\dagger & ww^\dagger \end{array}\right)$$

Measurement



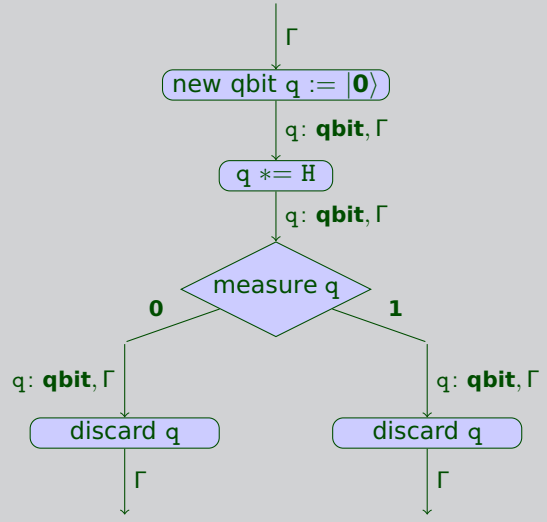
- let $|\varphi\rangle$ be a pure state, s.t.

$$|\varphi\rangle = \begin{pmatrix} v \\ w \end{pmatrix} \qquad |\varphi\rangle\langle\varphi| = \left(\begin{array}{c|c} vv^\dagger & vw^\dagger \\ \hline wv^\dagger & ww^\dagger \end{array}\right)$$

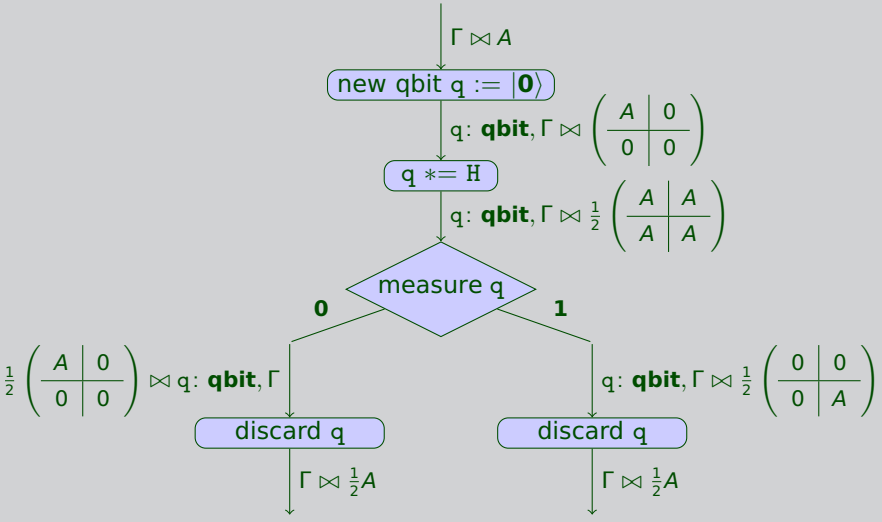
- measuring the first qubit of $|\varphi\rangle$ in the computational basis, yields

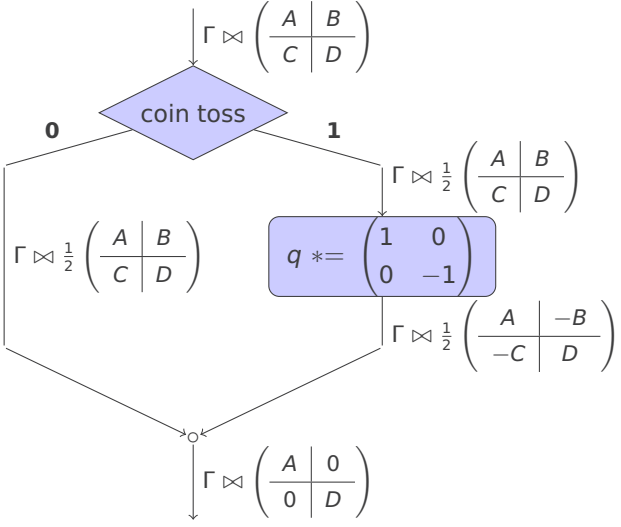
$$\|v\|^2 : \left(\begin{array}{c|c} vv^\dagger & 0 \\ \hline 0 & 0 \end{array}\right) \qquad \|w\|^2 : \left(\begin{array}{c|c} 0 & 0 \\ \hline 0 & ww^\dagger \end{array}\right)$$

Example (Coin Toss)

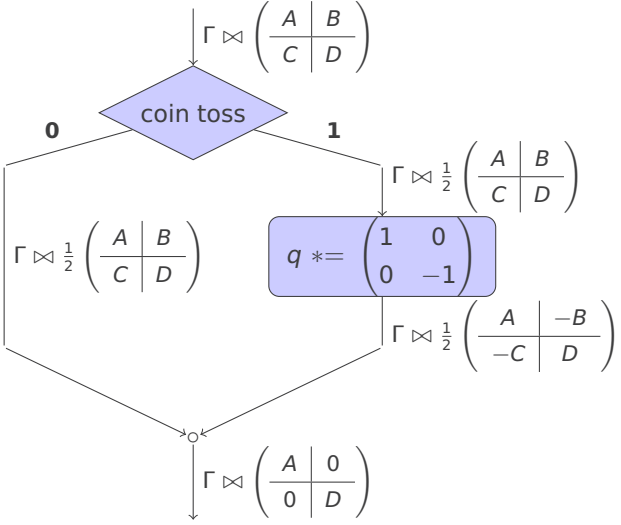


Example (Coin Toss)

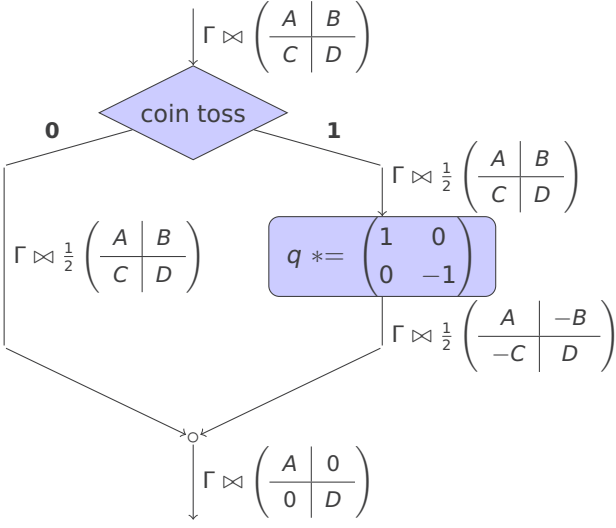


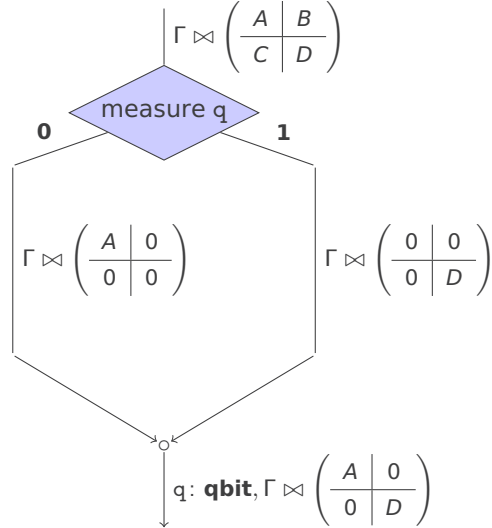

$\Gamma \bowtie \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right)$
coin toss
0
1
$\Gamma \bowtie \frac{1}{2} \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right)$
$q \mathrel{*}= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
$\Gamma \bowtie \frac{1}{2} \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right)$
$\Gamma \bowtie \frac{1}{2} \left(\begin{array}{c|c} A & -B \\ \hline -C & D \end{array}\right)$
$\Gamma \bowtie \left(\begin{array}{c|c} A & 0 \\ \hline 0 & D \end{array}\right)$

Semantical Program Equivalence

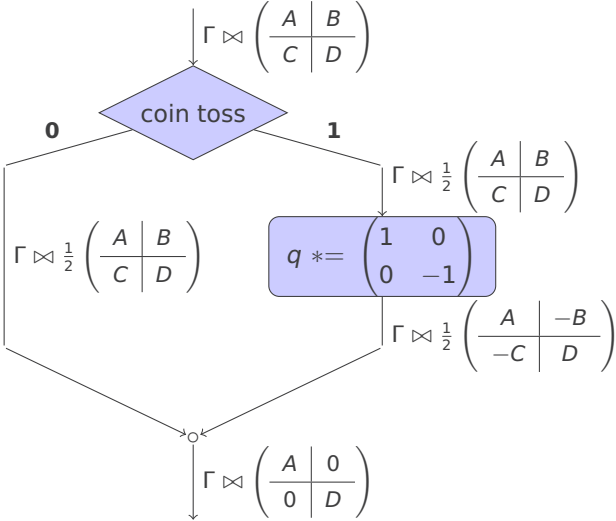


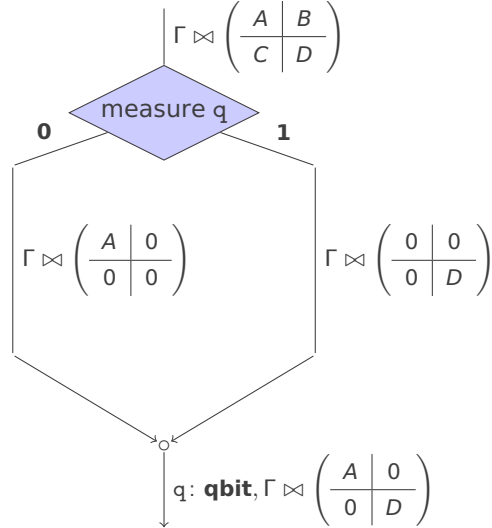
Semantical Program Equivalence



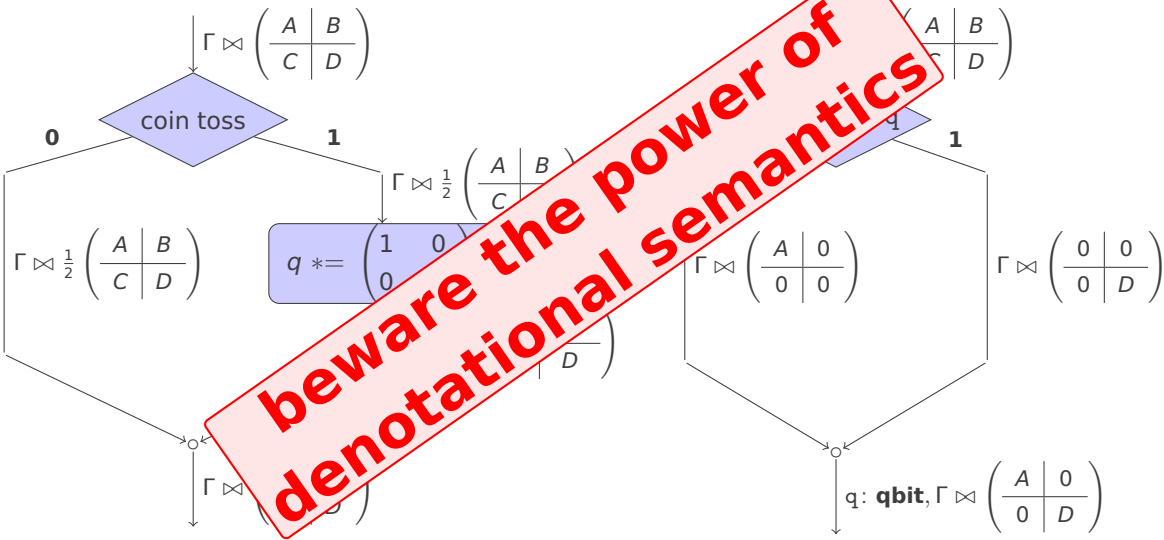


Semantical Program Equivalence: Program Optimisation





Semantical Program Equivalence: Program Optimisation



beware the power of denotational semantics

The Language QPL

QPL Terms

$$
\begin{aligned}
\texttt{stm}, \texttt{stm}_1, \texttt{stm}_2 \quad ::= \quad & \texttt{new bit b} := \mathbf{0} \mid \texttt{new qbit b} := |\mathbf{0}\rangle \mid \texttt{discard x} \\
& \mid \texttt{b} := \mathbf{0} \mid \texttt{b} := \mathbf{1} \mid \texttt{q}_1, \ldots, \texttt{q}_n \mathrel{*}= \texttt{S} \\
& \mid \texttt{skip} \mid \texttt{stm}_1 ; \texttt{stm}_2 \mid \\
& \mid \texttt{if}(\texttt{b}) \texttt{ then } \texttt{stm}_1 \texttt{ else } \texttt{stm}_2 \mid \texttt{measure q then } \texttt{stm}_1 \texttt{ else } \texttt{stm}_2 \\
& \mid \texttt{while b stm} \\
& \mid \texttt{proc } X \colon \Gamma \to \Gamma' \, \{\texttt{stm}_1\} \texttt{ in } \texttt{stm}_2 \mid \texttt{y}_1, \ldots, \texttt{y}_m := X(\texttt{x}_1, \ldots, \texttt{x}_n)
\end{aligned}
$$

The Language QPL

QPL Terms

$$
\begin{aligned}
\texttt{stm}, \texttt{stm}_1, \texttt{stm}_2 \quad ::= \quad & \texttt{new bit b}{:=}\mathbf{0} \mid \texttt{new qbit b}{:=}|\mathbf{0}\rangle \mid \texttt{discard x} \\
& \mid \texttt{b}{:=}\mathbf{0} \mid \texttt{b}{:=}\mathbf{1} \mid \texttt{q}_1, \ldots, \texttt{q}_n *{=} \texttt{S} \\
& \mid \texttt{skip} \mid \texttt{stm}_1; \texttt{stm}_2 \mid \\
& \mid \texttt{if(b) then stm}_1 \texttt{ else stm}_2 \mid \texttt{measure q then stm}_1 \texttt{ else stm}_2 \\
& \mid \texttt{while b stm} \\
& \mid \texttt{proc } X\colon \Gamma \to \Gamma' \{\texttt{stm}_1\} \texttt{ in stm}_2 \mid \texttt{y}_1, \ldots, \texttt{y}_m {=} X(\texttt{x}_1, \ldots, \texttt{x}_n)
\end{aligned}
$$

Comments

- $\texttt{proc } X\colon \Gamma \to \Gamma' \{\texttt{stm}_1\} \texttt{ in stm}_2$ defines a procedure with body $\texttt{stm}_1$ and scope $\texttt{stm}_2$
- $\texttt{y}_1, \ldots, \texttt{y}_m {=} X(\texttt{x}_1, \ldots, \texttt{x}_n)$ denotes a procedure call

The Language QPL

QPL Terms

$$
\begin{aligned}
\texttt{stm}, \texttt{stm}_1, \texttt{stm}_2 \quad ::= \quad & \texttt{new bit b}{:=}\mathbf{0} \mid \texttt{new qbit b}{:=}|\mathbf{0}\rangle \mid \texttt{discard x} \\
& \mid \texttt{b}{:=}\mathbf{0} \mid \texttt{b}{:=}\mathbf{1} \mid \texttt{q}_1, \ldots, \texttt{q}_n *{=} \texttt{S} \\
& \mid \texttt{skip} \mid \texttt{stm}_1; \texttt{stm}_2 \mid \\
& \mid \texttt{if(b) then stm}_1 \texttt{ else stm}_2 \mid \texttt{measure q then stm}_1 \texttt{ else stm}_2 \\
& \mid \texttt{while b stm} \\
& \mid \texttt{proc } X\colon \Gamma \to \Gamma' \{\texttt{stm}_1\} \texttt{ in stm}_2 \mid \texttt{y}_1, \ldots, \texttt{y}_m {=} X(\texttt{x}_1, \ldots, \texttt{x}_n)
\end{aligned}
$$

Comments

- $\texttt{proc } X\colon \Gamma \to \Gamma' \{\texttt{stm}_1\} \texttt{ in stm}_2$ defines a procedure with body $\texttt{stm}_1$ and scope $\texttt{stm}_2$
- $\texttt{y}_1, \ldots, \texttt{y}_m {=} X(\texttt{x}_1, \ldots, \texttt{x}_n)$ denotes a procedure call

Summary on QPL

- each edge in the quantum flow chart has been given an annotation in the form of a density matrix
- the semantics of a QPL program is now given as a mapping of the input density matrix to the output density matrix
- in order to show well-definedness these mapping are couched as morphisms of a category based on superoperators
- this yields a compositional denotational semantics

Summary on QPL

- each edge in the quantum flow chart has been given an annotation in the form of a density matrix
- the semantics of a QPL program is now given as a mapping of the input density matrix to the output density matrix
- in order to show well-definedness these mapping are couched as morphisms of a category based on superoperators
- this yields a compositional denotational semantics

Comparison

	Allocation	Subroutines	Oracles	Data Types	Specification	Resources
QPL	✓	✓	(✓)	✓	(✓)	×

Summary on QPL

- each edge in the quantum flow chart has been given an annotation in the form of a density matrix
- the semantics of a QPL program is now given as a mapping of the input density matrix to the output density matrix
- in order to show well-definedness these mapping are couched as morphisms of a category based on superoperators
- this yields a compositional denotational semantics

Comparison

	Allocation	Subroutines	Oracles	Data Types	Specification	Resources
QPL	✓	✓	(✓)	✓	(✓)	×
Quipper	✓	✓	✓	✓	(✓)	×

Summary on QPL

- each edge in the quantum flow chart has been given an annotation in the form of a density matrix
- the semantics of a QPL program is now given as a mapping of the input density matrix to the output density matrix
- in order to show well-definedness these mapping are couched as morphisms of a category based on superoperators
- this yields a compositional denotational semantics

Comparison

	Allocation	Subroutines	Oracles	Data Types	Specification	Resources
QPL	✓	✓	(✓)	✓	(✓)	×
Quipper	✓	✓	✓	✓	(✓)	×
Q#	✓	✓	✓	✓	×	×

universität
innsbruck

Program and Resource Analysis

Static Program Analysis

Definition (due to Z. Manna and A. Pnueli)

"The algorithmic discovery of properties of a program by inspection of the source text."

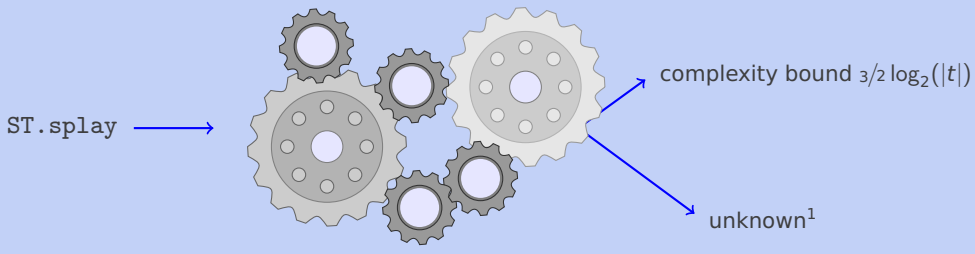
1

Static Program Analysis

Definition (due to Z. Manna and A. Pnueli)

"The algorithmic discovery of properties of a program by inspection of the source text."

Push-Button Automation



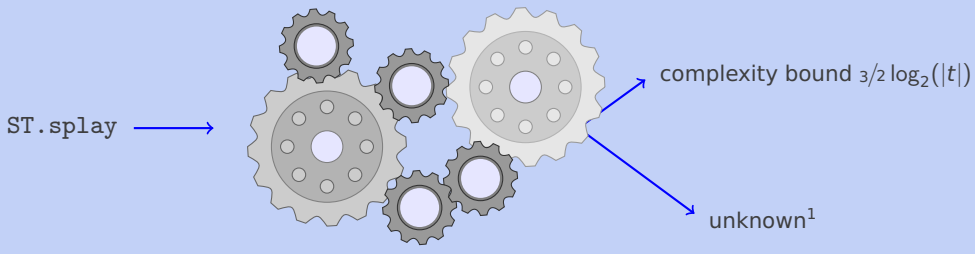
1

Static Program Analysis

Definition (due to Z. Manna and A. Pnueli)

"The algorithmic discovery of properties of a program by inspection of the source text."

Push-Button Automation



[1]The problem is (highly) undecidable, to be precise it is Σ_2^0-complete.

Zune Bug: December 31, 2008


```
year = ORIGINYEAR; /* = 1980 */
while (days > 365) {
  if (IsLeapYear(year)) {
    if (days > 366) {
      days -= 366;
      year += 1; } }
  else {
    days -= 365;
    year += 1; }
}
```

Zune Bug: December 31, 2008


```
year = ORIGINYEAR; /* = 1980 */
while (days > 365) {
  if (IsLeapYear(year)) ⌜
    if (days > 366) {
      days -= 366;
      year += 1; } }
  else {
    days -= 365;
    year += 1; }
}
```

Zune Bug: December 31, 2008


```
year = ORIGINYEAR; /* = 1980 */
while (days > 365) {
  if (IsLeapYear(year)) {
    if (days > 366) {
      days -= 366;
      year += 1; } }
  else {
    days -= 365;
    year += 1; }
}
```

D. Distefano, M. Fähndrich, F. Logozzo, and P. O'Hearn.
Scaling Static Analyses at Facebook. *Commun. ACM*, 62(8):62–70, 2019.

Zune Bug: December 31, 2008


```
year = ORIGINYEAR; /* = 1980 */
while (days > 365) {
  if (IsLeapYear(year)) {
    if (days > 366) {
      days -= 366;
      year += 1; } }
    else {
      days -= 365;
      year += 1; }
}
```

SOLVED!

D. Distefano, M. Fähndrich, F. Logozzo, and P. O'Hearn.
Scaling Static Analyses at Facebook. *Commun. ACM*, 62(8):62–70, 2019.

Quantum Bugs

M. Paltenghi and M. Pradel.
Bugs in quantum computing platforms: An empirical study. *CoRR*, abs/2110.14560, 2022.

Quantum Bugs

- empirical study of bugs in a broad range of quantum computing platforms (eg. IBM's Qiskit, Google's Circ, Mircosoft's Q#)
- research methodology: search for commit message like "fix" together with evidence of a commit that patched the problem
- 223 bugs found, among them (almost) 40% quantum-specific

M. Paltenghi and M. Pradel.
Bugs in quantum computing platforms: An empirical study. *CoRR*, abs/2110.14560, 2022.

Quantum Bugs

- empirical study of bugs in a broad range of quantum computing platforms (eg. IBM's Qiskit, Google's Circ, Mircosoft's Q#)
- research methodology: search for commit message like "fix" together with evidence of a commit that patched the problem
- 223 bugs found, among them (almost) 40% quantum-specific
- bugs occur everywhere, quantum-specific ones parts that represent, compile or optimise quantum code
- classical bugs occur in seemingly easy adminstrative code, like infrastructural scripts
- study proposes a hierarchy of bug patterns: ten quantum-specific patterns

M. Paltenghi and M. Pradel.
Bugs in quantum computing platforms: An empirical study. *CoRR*, abs/2110.14560, 2022.

Quantum-Specific Patterns

Incorrect Numerical Calculation

```
return pauli_string_phasor.PauliStringPhasor(
  PauliString(qubit_pauli_map=self._qubit_pauli_map),
    exponent_neg=+half_turns / 4,
    exponent_pos=-half_turns / 4)
```

Quantum-Specific Patterns

Incorrect Numerical Calculation

```
return pauli_string_phasor.PauliStringPhasor(
  PauliString(qubit_pauli_map=self._qubit_pauli_map),
    exponent_neg=+half_turns / 2,
    exponent_pos=-half_turns / 2)
```

Quantum-Specific Patterns

Incorrect Numerical Calculation

```
return pauli_string_phasor.PauliStringPhasor(
  PauliString(qubit_pauli_map=self._qubit_pauli_map),
    exponent_neg=+half_turns / 2,
    exponent_pos=-half_turns / 2)
```

Comments

- functional bugs clearly dominate non-functional bugs (like inefficient code)
- most likely bug symptoms are crashes or incorrect outputs; the latter are typically quantum-specific

Quantum-Specific Patterns

Incorrect Numerical Calculation

```
return pauli_string_phasor.PauliStringPhasor(
  PauliString(qubit_pauli_map=self._qubit_pauli_map),
    exponent_neg=+half_turns / 2,
    exponent_pos=-half_turns / 2)
```

Comments

- functional bugs clearly dominate non-functional bugs (like inefficient code)
- most likely bug symptoms are crashes or incorrect outputs; the latter are typically quantum-specific
- testing doesn't help
- compile-time guarantees through better type systems or specification languages would

Quantum-Specific Patterns

Incorrect Numerical Calculation

```
return pauli_string_phasor.PauliStringPhas
  PauliString(qubit_pauli_map=self._qub
    exponent_neg=+half_turns / 2,
    exponent_pos=-half_turns / 2)
```

not at all solved
formal methods required

Comments

- functional bugs clearl tional bugs (like inefficient code)
- most likely bug sy or incorrect outputs; the latter are typically quantum-spec
- testing do
- compile-ti through better type systems or specification languages would

universität
innsbruck

Expected Cost Analysis for Quantum Programs

Resource Analysis of Quantum Programs

resources could be

- the expected runtime
- the expected number of quantum gates
- the amount of quantum resources (in an application-specific sense) required by quantum programs,

Resource Analysis of Quantum Programs

resources could be

- the expected runtime
- the expected number of quantum gates
- the amount of quantum resources (in an application-specific sense) required by quantum programs,

Our Results

1. formal method for an expected cost analysis providing a framework for future (partial) automation

Resource Analysis of Quantum Programs

resources could be

- the expected runtime
- the expected number of quantum gates
- the amount of quantum resources (in an application-specific sense) required by quantum programs,

Our Results

1. formal method for an expected cost analysis providing a framework for future (partial) automation
2. computing the expected cost of several well-known quantum algorithms and protocols, such as
   - coin tossing
   - repeat until success
   - entangled state preparation
   - quantum walk

Resource Analysis of Quantum Programs

resources could be

- the expected runtime
- the expected number of quantum gates
- the amount of quantum resources (in an application-specific sense) required by quantum programs,

Our Results

1. formal method for an expected cost analysis providing a framework for future (partial) automation
2. computing the expected cost of several well-known quantum algorithms and protocols, such as
   - coin tossing
   - repeat until success
   - entangled state preparation
   - quantum walk

Coin Tossing: Syntax by Example

$$
\begin{array}{l}
CT(\mathtt{q}) \triangleq\ \mathtt{x} = \mathtt{true};\\
\qquad\quad \mathtt{while(x)\{}\\
\qquad\qquad \mathtt{q} \mathrel{*}= \mathtt{H};\\
\qquad\qquad \mathtt{x} = \mathtt{meas(q)};\\
\qquad\qquad \mathtt{consume(1)}\\
\qquad\qquad \}
\end{array}
\Bigg\}\mathtt{stm}_0
$$

- iterated coin toss, probability of termination after n steps depends on initial state $|\varphi\rangle$ of qubit q
- loop body $\mathtt{stm}_0$ consumes 1 resource
- overall probability of termination $= 1$

Coin Tossing: Syntax by Example

$$\mathtt{stm}, \mathtt{stm}_1, \mathtt{stm}_2 \quad ::= \quad \mathtt{skip} \mid \mathtt{x}^{\mathcal{K}} = \mathtt{e}^{\mathcal{K}} \mid \overline{\mathtt{q}} \mathrel{*}= \mathtt{U} \mid \mathtt{x}^{\mathcal{B}} = \mathtt{meas(q)} \mid \mathtt{consume(a)} \mid \mathtt{stm}_1; \mathtt{stm}_2 \mid \mathtt{if}\ (\mathtt{b})\ \{\mathtt{stm}_1\}\ \mathtt{else}\ \{\mathtt{stm}_2\} \mid \mathtt{while(b)}\ \{\mathtt{stm}\}$$

- a state σ is a pair $(s, |\varphi\rangle)$ consisting of a store s and a quantum state $|\varphi\rangle$
- a configuration μ is pair $(\mathtt{stm}, \sigma)$

$$\begin{array}{l} CT(\mathtt{q}) \triangleq\ \mathtt{x} = \mathtt{true}; \\ \quad\quad \mathtt{while(x)\{} \\ \quad\quad\quad \mathtt{q} \mathrel{*}= \mathtt{H}; \\ \quad\quad\quad \mathtt{x} = \mathtt{meas(q)}; \\ \quad\quad\quad \mathtt{consume(1)} \\ \quad\quad \mathtt{\}} \end{array} \Big\}\, \mathtt{stm}_0$$

- iterated coin toss, probability of termination after n steps depends on initial state $|\varphi\rangle$ of qubit $\mathtt{q}$
- loop body $\mathtt{stm}_0$ consumes 1 resource
- overall probability of termination $= 1$

Coin Tossing: Expected Cost Analysis

Manual Analysis

- let $(s, |\varphi\rangle)$ be a state, s.t.
  - $s(x) = \texttt{true}$
  - $|\varphi\rangle = \alpha\,|\mathbf{0}\rangle + \beta\,|\mathbf{1}\rangle$

$$\mathrm{ecost}_{CT(\mathsf{q})}(s, \begin{pmatrix}\alpha\\ \beta\end{pmatrix}) = \sup_{n\in\mathbb{N}} \left\{1 + p_1 \textstyle\sum_{i=0}^{n} \frac{1}{2^i}\right\} = 1 + |\alpha - \beta|^2$$

Coin Tossing: Expected Cost Analysis

Manual Analysis

- let $(s, |\varphi\rangle)$ be a state, s.t.
  - $s(x) = \texttt{true}$
  - $|\varphi\rangle = \alpha\,|\mathbf{0}\rangle + \beta\,|\mathbf{1}\rangle$

$$\mathrm{ecost}_{CT(\mathrm{q})}(s, \begin{pmatrix}\alpha\\ \beta\end{pmatrix}) = \sup_{n\in\mathbb{N}} \left\{1 + p_1 \textstyle\sum_{i=0}^{n} \frac{1}{2^i}\right\} = 1 + |\alpha - \beta|^2$$

- eg. $\mathrm{ecost}_{CT(\mathrm{q})}(s, |\mathbf{0}\rangle) = 2$

Coin Tossing: Expected Cost Analysis

Manual Analysis

- let $(s, |\varphi\rangle)$ be a state, s.t.
  - $s(x) = \texttt{true}$
  - $|\varphi\rangle = \alpha\,|\mathbf{0}\rangle + \beta\,|\mathbf{1}\rangle$

$$\mathrm{ecost}_{CT(\mathtt{q})}(s, \begin{pmatrix}\alpha\\ \beta\end{pmatrix}) = \sup_{n\in\mathbb{N}} \left\{1 + p_1 \textstyle\sum_{i=0}^{n} \frac{1}{2^i}\right\} = 1 + |\alpha - \beta|^2$$

- eg. $\mathrm{ecost}_{CT(\mathtt{q})}(s, |\mathbf{0}\rangle) = 2$

requires

- formal operational semantics (roughly $[\![\mathtt{stm}]\!] \colon \mathtt{State} \to \mathcal{D}(\mathtt{State})$)
- semantics accounts for resource consumption
- but tedious, error-prone calculations!

Quantum Expected Cost Transformer

(generalisation of Hoare-style formal verification)

$$\mathtt{qect}\left[\cdot\right]\left\{\cdot\right\} : \mathtt{Program} \to (\mathtt{State} \to \mathbb{R}^{+\infty}) \to (\mathtt{State} \to \mathbb{R}^{+\infty})$$

Quantum Expected Cost Transformer

(generalisation of Hoare-style formal verification)

$$\mathtt{qect}\left[\cdot\right]\left\{\cdot\right\} : \mathtt{Program} \to (\mathtt{State} \to \mathbb{R}^{+\infty}) \to (\mathtt{State} \to \mathbb{R}^{+\infty})$$

resources (expected costs)
available after execution

Quantum Expected Cost Transformer

(generalisation of Hoare-style formal verification)

resources required before execution

$$\mathtt{qect}\left[\cdot\right]\left\{\cdot\right\} : \mathtt{Program} \to (\mathtt{State} \to \mathbb{R}^{+\infty}) \to (\mathtt{State} \to \mathbb{R}^{+\infty})$$

resources (expected costs) available after execution

Quantum Expected Cost Transformer

(generalisation of Hoare-style formal verification)

resources required before execution

$$\mathtt{qect}\left[\cdot\right]\left\{\cdot\right\} : \mathtt{Program} \to (\mathtt{State} \to \mathbb{R}^{+\infty}) \to (\mathtt{State} \to \mathbb{R}^{+\infty})$$

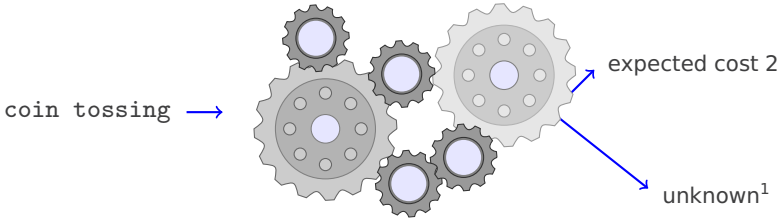
resources (expected costs) available after execution

Theorem

The following identity hold, for all $\mathtt{stm} \in \mathtt{Statement}$ *and* $\sigma \in \mathtt{State}$

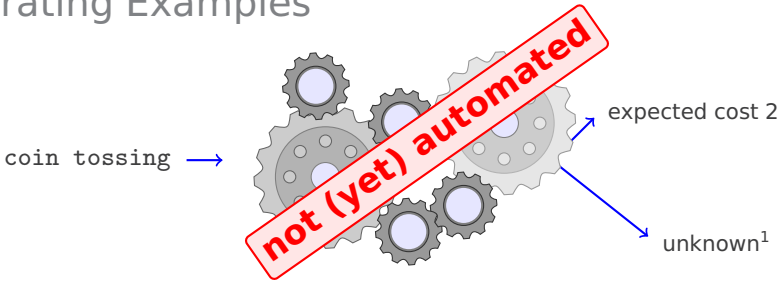
$$\mathtt{qect}\left[\mathtt{stm}\right]\left\{\underline{0}\right\}(\sigma) = \mathrm{ecost}_{\mathtt{stm}}(\sigma)$$

Illustrating Examples



`stm`	cost metric	expect cost
coin tossing	# of loops	2
repeat-until-success	# of T gates	$\frac{8}{3}$
entangled state preparation	# of attempts for k qubits	$148 \cdot (k+4)$
quantum walk on n-circle	# of loops	2 if started at position $\lvert\mathbf{1}\rangle$ 1 if started at position $\lvert\mathbf{0}\rangle$

Illustrating Examples



stm	cost metric	expect cost
coin tossing	# of loops	2
repeat-until-success	# of T gates	$\frac{8}{3}$
entangled state preparation	# of attempts for k qubits	$148 \cdot (k+4)$
quantum walk on n-circle	# of loops	2 if started at position $\|\mathbf{1}\rangle$ 1 if started at position $\|\mathbf{0}\rangle$

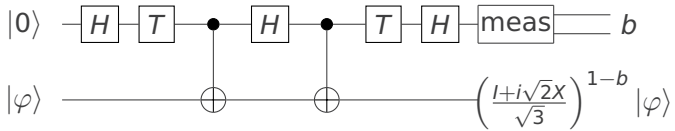
Illustrating Examples

$CT(\mathtt{q}) \triangleq$
```
x = true;
while(x){
  q *= H;
  x = meas(q);
  consume(1)
  }
```
} $\mathtt{stm}_0$

- iterated coin toss
- loop body $\mathtt{stm}_0$ consumes 1 resource
- overall probability of termination $= 1$

stm	cost metric	expect cost
coin tossing	# of loops	2
repeat-until-success	# of T gates	$\frac{8}{3}$
entangled state preparation	# of attempts for k qubits	$148 \cdot (k+4)$
quantum walk on n-circle	# of loops	2 if started at position $\lvert\mathbf{1}\rangle$ 1 if started at position $\lvert\mathbf{0}\rangle$

Illustrating Examples



stm	cost metric	expect cost
coin tossing	# of loops	2
repeat-until-success	# of T gates	$\frac{8}{3}$
entangled state preparation	# of attempts for k qubits	$148 \cdot (k+4)$
quantum walk on n-circle	# of loops	2 if started at position $\|\mathbf{1}\rangle$ 1 if started at position $\|\mathbf{0}\rangle$

Illustrating Examples

$CHAIN(k, q_0, \ldots, q_{k+3}) \triangleq$

```
  ...
  while(0 ≤ t ∧ t < k){
    CHAIN4(q_{t+1}, q_{t+2}, q_{t+3}, q_{t+4});
    FUSE(q_t, q_{t+1}, x);
    if(x){t = t + 4} else {t = t − 1};
    if(t = −1){t = 0; q_0 = |+⟩} else {skip}
  }
```

- prepare a graph state represented by a path on k qubits
- implemented with nested while loops

stm	cost metric	expect cost
coin tossing	# of loops	2
repeat-until-success	# of T gates	$\frac{8}{3}$
entangled state preparation	# of attempts for k qubits	$148 \cdot (k+4)$
quantum walk on n-circle	# of loops	2 if started at position $\|\mathbf{1}\rangle$ 1 if started at position $\|\mathbf{0}\rangle$

Illustrating Examples

```
x^B = true;
while(x){
  x = meas(p);
  q *= H;
  q,p *= S;
  consume(1)
}
```

- Hadamard quantum walk on an n-circle (Liu et al., 2019)
- cost analysis depends on quantum state
- loop body $\texttt{stm}_0$ consumes 1 resource

stm	cost metric	expect cost
coin tossing	# of loops	2
repeat-until-success	# of T gates	$\frac{8}{3}$
entangled state preparation	# of attempts for k qubits	$148 \cdot (k+4)$
quantum walk on n-circle	# of loops	2 if started at position $\lvert\mathbf{1}\rangle$ 1 if started at position $\lvert\mathbf{0}\rangle$

Illustrating Examples

```
x^B = true;
while(x){
  x = meas(p);
  q *= H;
  q,p *= S;
  consume(1)
}
```

$\Sigma_{i=0}^{n-1} |L\rangle\langle L| \otimes |i \ominus 1\rangle\langle i| + \Sigma_{i=0}^{n-1} |R\rangle\langle R| \otimes |i \oplus 1\rangle\langle i|$

- Hadamard quantum walk on an n-circle (Liu et al., 2019)
- cost analysis depends on quantum state
- loop body $\texttt{stm}_0$ consumes 1 resource

stm	cost metric	expect cost
coin tossing	# of loops	2
repeat-until-success	# of T gates	$\frac{8}{3}$
entangled state preparation	# of attempts for k qubits	$148 \cdot (k+4)$
quantum walk on n-circle	# of loops	2 if started at position $\|\mathbf{1}\rangle$ 1 if started at position $\|\mathbf{0}\rangle$

Summary on Resource Analysis of Quantum Programs

- denotational semantics as a variant of the quantum expected cost transformer, employing density matrices

Summary on Resource Analysis of Quantum Programs

- denotational semantics as a variant of the quantum expected cost transformer, employing density matrices

$$\mathtt{qev}_{\mathtt{K}}\left[\!\left[\cdot\right]\!\right]\left\{\!\left\{\cdot\right\}\!\right\} : \mathtt{Program} \to (\mathtt{State} \to \mathtt{K}) \to (\mathtt{State} \to \mathtt{K})$$

Summary on Resource Analysis of Quantum Programs

- denotational semantics as a variant of the quantum expected cost transformer, employing density matrices

$$\{0,1\}^n \times \mathbb{Z}^m \to D_{2^k}$$

$$\mathtt{qev}_{\mathtt{K}}\left[\cdot\right]\left\{\cdot\right\} : \mathtt{Program} \to (\mathtt{State} \to \mathtt{K}) \to (\mathtt{State} \to \mathtt{K})$$

Summary on Resource Analysis of Quantum Programs

- denotational semantics as a variant of the quantum expected cost transformer, employing density matrices

$$\{0,1\}^n \times \mathbb{Z}^m \to D_{2^k}$$

$$\mathtt{qev}_{\mathtt{K}}\left[\cdot\right]\left\{\cdot\right\} : \mathtt{Program} \to (\mathtt{State} \to \mathtt{K}) \to (\mathtt{State} \to \mathtt{K})$$

- (partial) automation subject of future work

Summary on Resource Analysis of Quantum Programs

- denotational semantics as a variant of the quantum expected cost transformer, employing density matrices

$$\{0,1\}^n \times \mathbb{Z}^m \to D_{2^k}$$

$$\mathrm{qev}_{\mathrm{K}}\left[\cdot\right]\left\{\cdot\right\} : \mathtt{Program} \to (\mathtt{State} \to \mathrm{K}) \to (\mathtt{State} \to \mathrm{K})$$

- (partial) automation subject of future work

Comparison

	Allocation	Subroutines	Oracles	Data Types	Specification	Resources
qWhile	✓	×	×	(✓)	(✓)	✓

M. Avanzini, G. Moser, R. Péchoux, S. Perdrix, and V. Zamdzhiev.
Quantum expectation transformers for cost analysis. *CoRR*, abs/2201.09361, 2022.

Thank You For Your Attention!

Quantum Expectation Transformer

$\mathtt{stm}$	$\mathtt{qet}\left[\mathtt{stm}\right]\left\{f\right\}$
ϵ	f
$\mathtt{skip}$	f
$\mathtt{x} = \mathtt{e}$	$f[\mathtt{x} := \mathtt{e}]$
$\overline{\mathtt{q}} *= \mathtt{U}$	$f[\mathtt{U}_{\overline{\mathtt{q}}}]$
$\mathtt{x} = \mathtt{meas}(\mathtt{q})$	$p_0^{\mathtt{q}} f[\mathtt{x} := 0; \mathtt{M}_0^{\mathtt{q}}] + (1 - p_0^{\mathtt{q}}) f[\mathtt{x} := 1; \mathtt{M}_1^{\mathtt{q}}]$
$\mathtt{consume}(\mathtt{a})$	$\max(\llbracket \mathtt{a} \rrbracket, \underline{0}) + f$
$\mathtt{stm}_1; \mathtt{stm}_2$	$\mathtt{qet}\left[\mathtt{stm}_1\right]\left\{\mathtt{qet}\left[\mathtt{stm}_2\right]\left\{f\right\}\right\}$
$\mathtt{if}(\mathtt{b})\{\mathtt{stm}_1\}\ \mathtt{else}\ \{\mathtt{stm}_2\}$	$\mathtt{qet}\left[\mathtt{stm}_1\right]\left\{f\right\} +_{\llbracket \mathtt{b} \rrbracket} \mathtt{qet}\left[\mathtt{stm}_2\right]\left\{f\right\}$
$\mathtt{while}(\mathtt{b})\{\mathtt{stm}\}$	$\mathrm{lfp}\left(\lambda F.\mathtt{qet}\left[\mathtt{stm}\right]\left\{F\right\} +_{\llbracket \mathtt{b} \rrbracket} f\right)$

Coin Tossing: Formal Expected Cost Analysis

- we need to find an expectation g satisfying the following inequalities

$$[\![\neg \mathtt{x}]\!] \cdot \underline{0} \leqslant g$$
$$[\![\mathtt{x}]\!] \cdot \mathtt{qect}\Big[\mathtt{stm}_0\Big]\Big\{g\Big\} \leqslant g$$

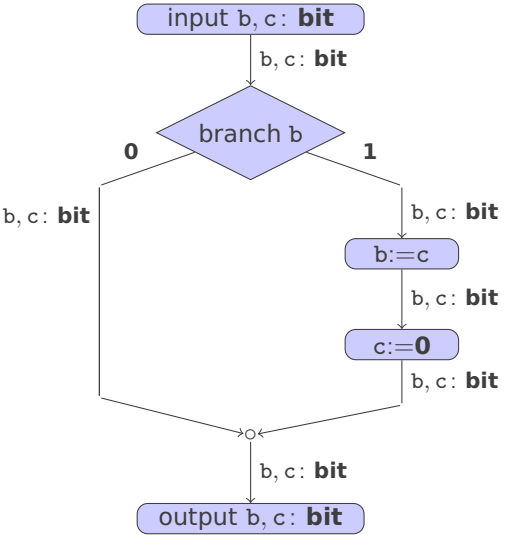
- we set

$$g(s, \begin{pmatrix} \alpha \\ \beta \end{pmatrix}) = [\![\mathtt{x}]\!] \cdot (1 + |\alpha - \beta|^2).$$

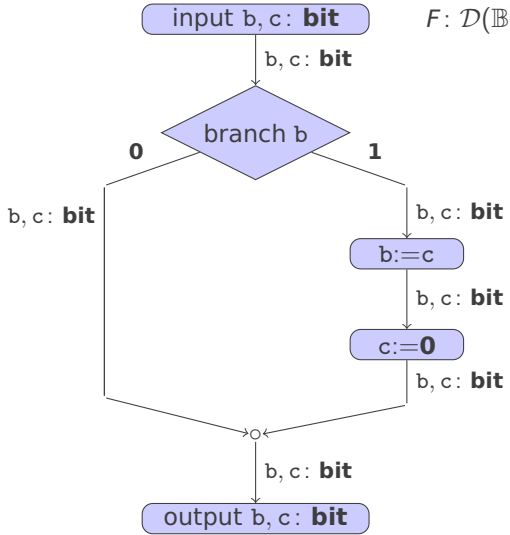
- expectation q satisfies the above inequality and we obtain

$$\mathtt{qect}\Big[CT(\mathtt{q})\Big]\Big\{\underline{0}\Big\} \leqslant \mathtt{qect}\Big[\mathtt{x}^{\mathcal{B}} = \mathtt{true}\Big]\Big\{g\Big\}$$
$$= g[\mathtt{x} := 1] = \lambda(s, \begin{pmatrix} \alpha \\ \beta \end{pmatrix}).1 + |\alpha - \beta|^2$$

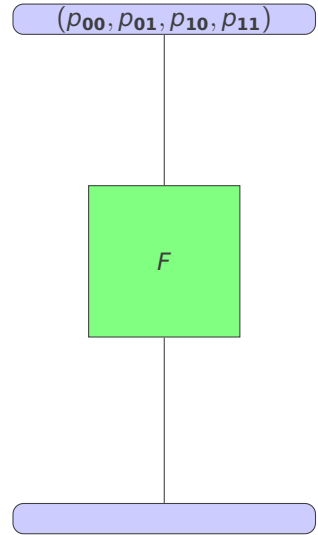
QPL: Classic Part



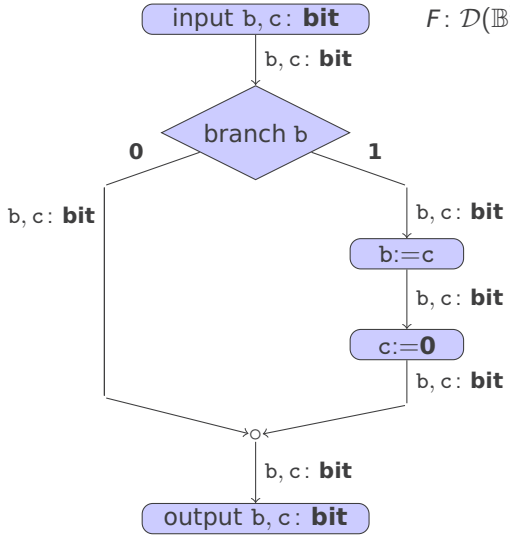
QPL: Classic Part



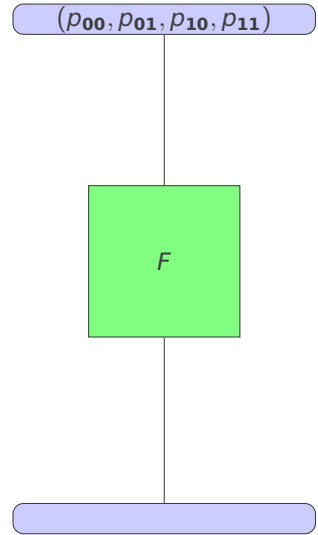
$F\colon \mathcal{D}(\mathbb{B}^2) \to \mathcal{D}(\mathbb{B}^2)$



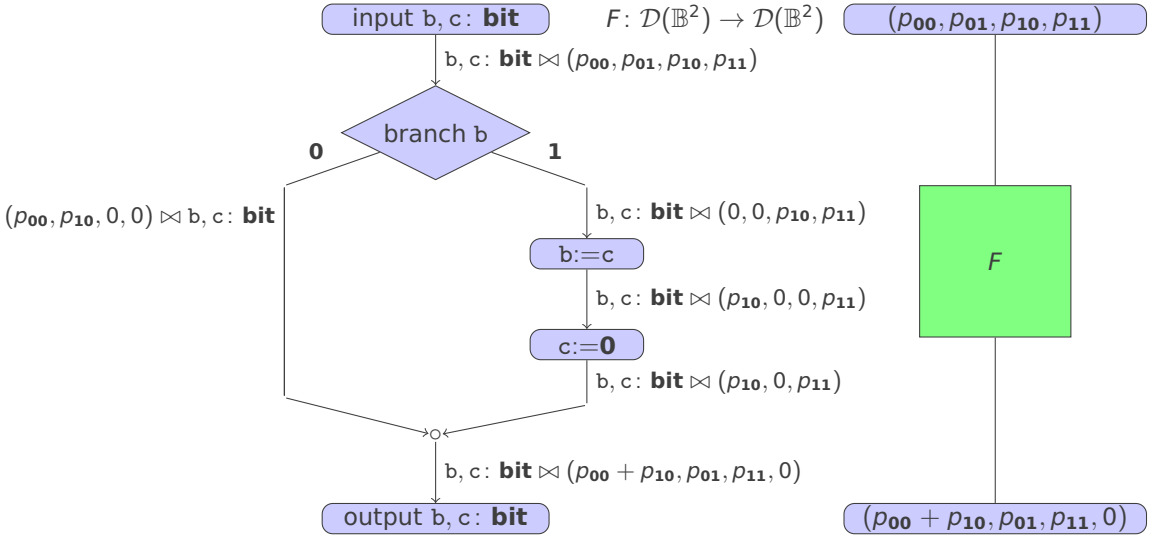
QPL: Classic Part



$F\colon \mathcal{D}(\mathbb{B}^2) \to \mathcal{D}(\mathbb{B}^2)$

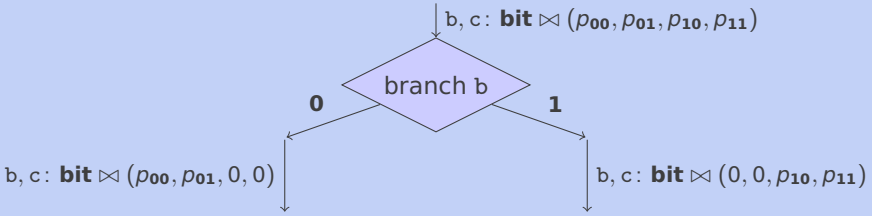


QPL: Classic Part



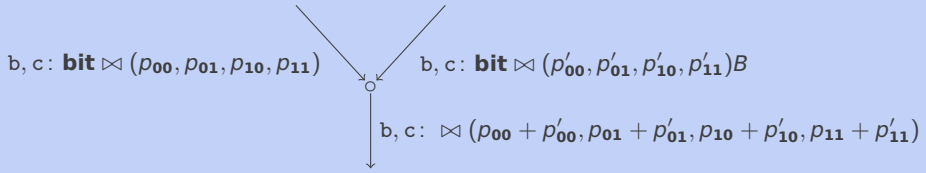
Classical Flow Charts

Branching



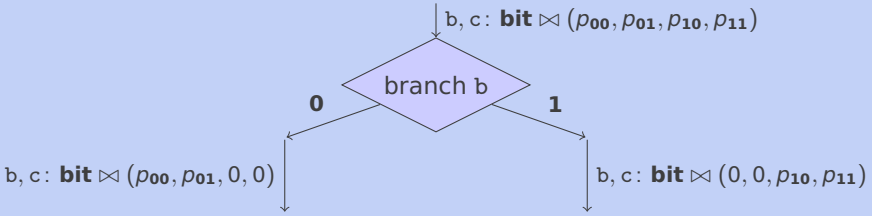
Classical Flow Charts

Merge



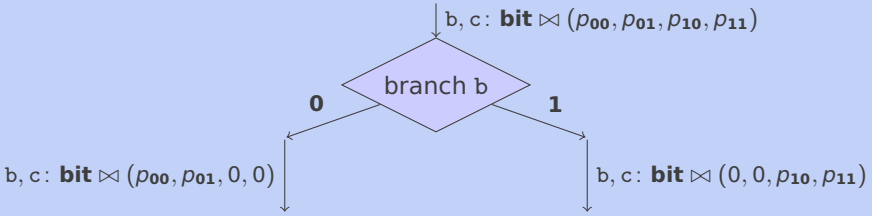
Classical Flow Charts

Branching



Classical Flow Charts

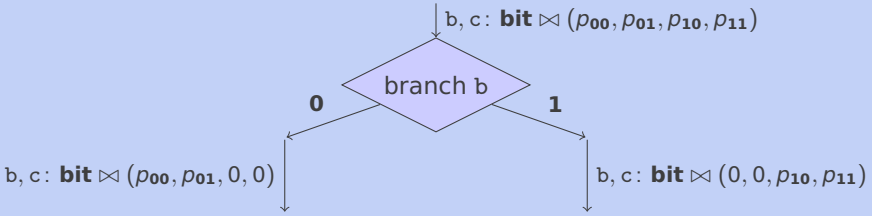
Branching



- a state of a program is a pair (e, σ)
  - e is an edge (conceivable as program location)
  - $\sigma : \mathcal{V} \to \mathbb{B}$ is a store

Classical Flow Charts

Branching



- a state of a program is a pair (e, σ)
  - e is an edge (conceivable as program location)
  - $\sigma : \mathcal{V} \to \mathbb{B}$ is a store
- observe that program locations and stores are interchangable:
  - program locations (aka program counters) are often just variables
  - different stores are representable by different control paths

Density Matrices & Löwner partial order

Definition

a density matrix is a positive Hermitian matrix A s.t. $tr(A) = 1$; a subdensity matrix is a positive Hermitian matrix A s.t. $tr(A) \leqslant 1$

Example

consider quantum state $|\varphi\rangle = \frac{1}{\sqrt{2}}|\mathbf{0}\rangle - \frac{1}{\sqrt{2}}|\mathbf{1}\rangle$, which is representable by the density matrix $|\varphi\rangle\langle\varphi|$:

$$|\varphi\rangle\langle\varphi| = \begin{pmatrix} 1/2 & -1/2 \\ -1/2 & 1/2 \end{pmatrix}$$

let D_n denote the set of subdensity matrices of dimension n

Definition

for matrices $A, B \in D_n$, define $A \sqsubset B$, if $B - A$ is positive

Proposition

the partial order set $(D_n, \sqsupseteq)$ is a CPO, that is, every increasing sequence admits least upper bounds