# On the Hardness of Analyzing Quantum Programs Quantitatively

Martin Avanzini[‡], Georg Moser[*], **Romain Péchoux**[†] and Simon Perdrix[†]

‡ Inria team Focus - Inria Côte d'Azur

∗ University of Innsbruck

† Inria team Mocqua - CNRS, Inria, Université de Lorraine - LORIA

ESOP24

April 10th, 2024




# Program verification and quantitative analysis

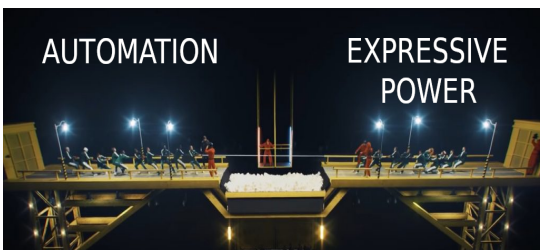
Study and design methods for **statically** analyzing the "**resource**" usage of programs:

- time, space,
- number of communications, energy, ...

Finding a balance between:

- **automation** (decidability, tractability)
- **expressive power** (false negatives)

A consequence of Rice's Theorem...



# Quantum programs

Quantum programs:

- have gained the focus due to the **quantum advantage** (Grover, Shor, ...)
- act on **qubits** $|\phi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, with $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$
- have a **measurement** construct yielding some probabilistic behavior:

$$\mathtt{x} := \mathtt{meas}\ \ |\phi\rangle \rightsquigarrow \begin{cases} \mathtt{x} := \mathtt{false} & \text{with prob. } |\alpha|^2 \\ \mathtt{x} := \mathtt{true} & \text{with prob. } |\beta|^2 \end{cases}$$

**Quantum states** are seen as objects $\binom{\alpha}{\beta}$ in $\mathbb{C}^2$, taking $|0\rangle = \binom{1}{0}$ and $|1\rangle = \binom{0}{1}$

Programs over $n$ qubits are seen as acting on the **Hilbert space** $\mathbb{C}^{2^n} = \underbrace{\mathbb{C}^2 \otimes \ldots \otimes \mathbb{C}^2}_{n \text{ times}}$

Classical-quantum programs are over $\mathtt{State} \triangleq \underbrace{\mathtt{Store} \times \mathcal{D}(\mathbb{C}^{2^n})}_{\text{Classical data} \times \text{Density operator}}$

# Static analysis of quantum programs: a difficult task

There are many **difficulties** to face towards automation:

- quantum programs are **not obvious**
  - low-level models like *quantum circuits* are more mainstream
- *quantum mechanics* yield **many constraints**:
  - *no-cloning* Theorem (impose linearity)
  - *quantum interference* (amplitudes may cancel out)
  - *entanglement* (difficulty to approximate)
- quantum programs work on Hilbert spaces of **complex numbers**:
  - *continuous* domain (equality is not decidable)
  - includes *non-computable* numbers

# Simple illustrative example: Cointoss(q)

```
x := true;
i := 0;
while x do {
    i := i + 1;
    q *= H;
    x := meas q
}
```

with $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$

# Simple illustrative example: Cointoss(q)

```
x := true;
i := 0;
while x do {
    i := i + 1;
    q *= H;
ℓ :  x := meas q
}
```

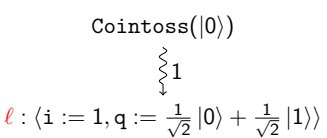
with $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$

Cointoss($|0\rangle$)

# Simple illustrative example: Cointoss(q)

$\mathtt{x} := \mathtt{true};$
$\mathtt{i} := 0;$
$\mathtt{while\ x\ do}\ \{$
$\quad \mathtt{i} := \mathtt{i} + 1;$
$\quad \mathtt{q} *= H;$
$\ell: \quad \mathtt{x} := \mathtt{meas\ q}$
$\}$

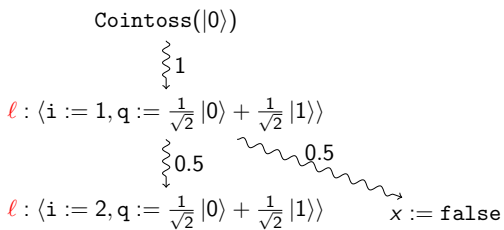
with $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$



# Simple illustrative example: Cointoss(q)

```
x := true;
i := 0;
while x do {
    i := i + 1;
    q *= H;
ℓ :  x := meas q
}
```

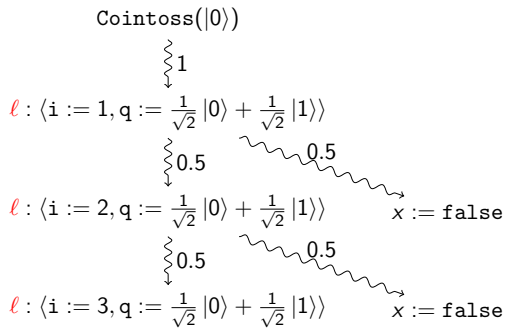
with $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$



# Simple illustrative example: Cointoss(q)

```
x := true;
i := 0;
while x do {
    i := i + 1;
    q *= H;
ℓ:  x := meas q
}
```

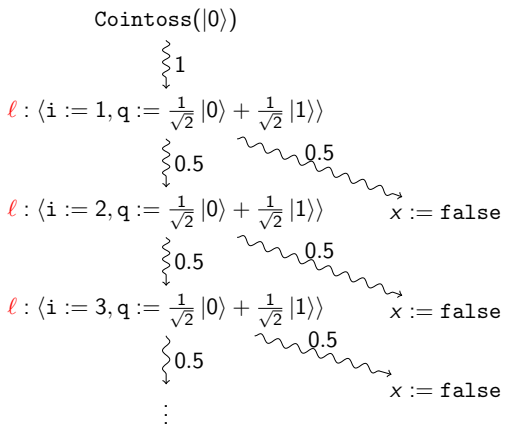
with $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$



# Simple illustrative example: Cointoss(q)

```
x := true;
i := 0;
while x do {
    i := i + 1;
    q *= H;
ℓ:  x := meas q
}
```

with $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$



# Quantitative properties of interest

- **Almost-Sure Termination** ($\textsc{Ast}$)
  - termination with prob. 1
- **Positive Almost-Sure Termination** ($\textsc{Past}$)
  - AST and finite mean derivation length
- Test the **Expected value** of a given variable ($\textsc{Test}_{=}$)
  - computed wrt a given *pre-expectation* $f \in \texttt{State} \to \mathbb{R}^+$
  - Find **upper and lower-bounds** on the expected value, e.g., ($\textsc{Test}_{<}$), ($\textsc{Test}_{>}$)
  - Test the **finiteness** of the expected value ($\textsc{Test}_{\neq\infty}$)
- All the above problems can be extended to their **Universal problem**
  - universal quantification over the input state, e.g., $\textsc{UPast}$, $\textsc{UTest}_{=}$

# Cointoss as an illustrating example

```
    x := true;
    i := 0;
    while x do {
        i := i + 1;
        q *= H;
ℓ :     x := meas q
    }
```

- $\texttt{Cointoss} \in \mathrm{UAST}$ (only the 1st iteration differs):

$$\ell : (\texttt{x} := \texttt{true}, \texttt{q} := \begin{pmatrix} \frac{\alpha+\beta}{\sqrt{2}} \\ \frac{\alpha-\beta}{\sqrt{2}} \end{pmatrix}) \rightsquigarrow_{|\frac{\alpha-\beta}{\sqrt{2}}|^2} (\texttt{x} := \texttt{true}, \texttt{q} := |1\rangle)$$

For $f \in \texttt{State} \to \mathbb{R}^+$ defined by $f(s, \rho) \triangleq s(\texttt{i})$:

- $(\texttt{Cointoss}, f, \lambda\langle s, \binom{\alpha}{\beta}\rangle.2-(\alpha\overline{\beta}+\overline{\alpha}\beta)) \in \mathrm{UTEST}_{=}$
- It also implies
  - $(\texttt{Cointoss}, f) \in \mathrm{UTEST}_{\neq\infty}$
  - $\texttt{Cointoss} \in \mathrm{UPAST}$

# Two restrictions

1. Restriction to **Clifford+T** gates:

$$I \triangleq \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right),\ X \triangleq \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right),\ Y \triangleq \left(\begin{smallmatrix} 0 & -i \\ i & 0 \end{smallmatrix}\right),\ Z \triangleq \left(\begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix}\right),\ H \triangleq \frac{1}{\sqrt{2}}\left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right),$$

$$S \triangleq \frac{1}{\sqrt{2}}\left(\begin{smallmatrix} 1 & 0 \\ 0 & i \end{smallmatrix}\right),\ CNOT \triangleq \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix},\ T \triangleq \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix}.$$

   - an **approximately universal** fragment of quantum mechanics
   - the state space consists of **algebraic complex numbers** $\overline{\mathbb{Q}}$
   - $=$, $\neq$, $\times$, $+$ are **decidable** over $\overline{\mathbb{Q}}$

2. Restriction to pre-expectations in $\{f \mid f : \texttt{State} \to \mathbb{R}^+ \cap \overline{\mathbb{Q}},\ f \text{ computable}\}$

# Reminder on the arithmetical hierarchy

## Definition [Levels of the arithmetical hierarchy]

$$\Pi_0^0 = \Sigma_0^0 \triangleq \mathrm{REC} \quad (\textit{decidable problems/recursive sets})$$
$$\Pi_{n+1}^0 \triangleq \{\psi \mid \exists \phi \in \Sigma_n^0,\ \forall \overline{x}.(\psi(\overline{x}) \iff \forall \overline{y}.\phi(\overline{x},\overline{y}))\}$$
$$\Sigma_{n+1}^0 \triangleq \{\psi \mid \exists \phi \in \Pi_n^0,\ \forall \overline{x}.(\psi(\overline{x}) \iff \exists \overline{y}.\phi(\overline{x},\overline{y}))\}$$

Examples of well-known (classical) complete problems:

$$\mathit{Halt}(p,d) \iff \exists t, p(d)\downarrow^t \text{ is } \Sigma_1^0 - \text{complete}$$
$$\mathit{CoHalt}(p,d) \iff \neg\exists t, p(d)\downarrow^t \text{ is } \Pi_1^0 - \text{complete}$$
$$\mathit{UHalt}(p) \iff \forall d,\ \exists t, p(d)\downarrow^t \text{ is } \Pi_2^0 - \text{complete}$$
$$\mathit{UCoHalt}(p) \iff \forall d,\ \nexists t, p(d)\downarrow^t \text{ is } \Pi_1^0 - \text{complete}$$
$$\mathit{UPolyHalt}(p) \iff \exists Q \in \mathbb{N}[X],\ \forall d,\ p(d)\downarrow^{Q(|d|)} \text{ is } \Sigma_2^0 - \text{complete}$$

## Completeness results over Clifford+T

| | Standard | | | Universal | | |
|---|---|---|---|---|---|---|
| *Expectation* | $\mathrm{TEST}_{>}$ | $\Sigma_1^0$ | | $\mathrm{UTEST}_{>}$ | $\Pi_2^0$ | (*) |
| | $\mathrm{TEST}_{\geq}$ | $\Pi_2^0$ | (*) | $\mathrm{UTEST}_{\geq}$ | $\Pi_2^0$ | (*) |
| | $\mathrm{TEST}_{=}$ | $\Pi_2^0$ | | $\mathrm{UTEST}_{=}$ | $\Pi_2^0$ | (*) |
| | $\mathrm{TEST}_{\leq}$ | $\Pi_1^0$ | (*) | $\mathrm{UTEST}_{\leq}$ | $\Pi_1^0$ | (*) |
| | $\mathrm{TEST}_{<}$ | $\Sigma_2^0$ | | $\mathrm{UTEST}_{<}$ | $\Pi_3^0$ | (*) |
| *Finiteness* | $\mathrm{TEST}_{\neq\infty}$ | $\Sigma_2^0$ | | $\mathrm{UTEST}_{\neq\infty}$ | $\Pi_3^0$ | (*) |
| *Termination* | AST | $\Pi_2^0$ | | UAST | $\Pi_2^0$ | |
| | PAST | $\Sigma_2^0$ | | UPAST | $\Pi_3^0$ | |

(*): new results not treated in the probabilistic case

Other results are surprising extensions of [Schnabl et al.2011], [Kaminski et al.2016] to the quantum setting

# A focus on Almost-Sure Termination

## Theorem

AST is $\Pi_2^0$-complete.

$|term_p^{\leq n}(d)| \triangleq$ the probability that $p(d)$ terminates in at most $n$ steps.
$|term_p(d)| \triangleq sup_{n\to\infty}|term_p^{\leq n}(d)|$.

$$\begin{aligned}(p,d) \in \text{AST} &\iff |term_p(d)| = 1 \\ &\iff \forall \epsilon \in \overline{\mathbb{Q}} \cap \mathbb{R}^+,\ \exists n \in \mathbb{N},\quad \underbrace{|term_p^{\leq n}(d)| \geq 1-\epsilon}_{\Pi_0^0 \text{ within the Clifford+T fragment}}\end{aligned}$$

Hardness is obtained by a reduction to *UHalt*.

## Corollary

UAST *is* $\Pi_2^0$*-complete.*

# Interesting but unsatisfactory results

Can we obtain **decidability** on the inference of such quantitative properties?

A solution is to use **quantum expectation transformers** on Clifford+T.

Historically:

- *Predicate transformers* for imperative programs
  - [Dijkstra76, Kozen85]
  - $\mathtt{pt} \in \mathtt{Prog} \to (\mathtt{Store} \to \{0,1\}) \to (\mathtt{Store} \to \{0,1\})$
- *Expectation transformers* for probabilistic programs
  - [McIver-Morgan05, Gretz-Katoen-McIver14, Kaminski-Katoen17]
  - $\mathtt{wp} \in \mathtt{Prog} \to (\mathtt{Store} \to \mathbb{R}^{+\infty}) \to (\mathtt{Store} \to \mathbb{R}^{+\infty})$
- *Quantum expectation transformers* for quantum programs
  - [Avanzini et al.22, Liu et al.22]
  - $\mathtt{qet} \in \mathtt{Prog} \to (\mathtt{State} \to \mathbb{R}^{+\infty}) \to (\mathtt{State} \to \mathbb{R}^{+\infty})$

# Quantum expectation transformers

**Definition** [$\mathtt{qet} \in \mathtt{Prog} \to (\mathtt{State} \to \mathbb{R}^{+\infty}) \to (\mathtt{State} \to \mathbb{R}^{+\infty})$]

$$\mathtt{qet}[\mathtt{x} := \mathtt{e}]\{f\} \triangleq \lambda(s,\rho).f(s[\mathtt{x} := [\![\mathtt{e}]\!]],\rho)$$
$$\mathtt{qet}[\mathtt{stm}_1\mathtt{;}\ \mathtt{stm}_2]\{f\} \triangleq \mathtt{qet}[\mathtt{stm}_1]\{\mathtt{qet}[\mathtt{stm}_2]\{f\}\}$$
$$\mathtt{qet}[\mathtt{while\ b\ do\ stm}]\{f\} \triangleq \mathsf{lfp}\left(\lambda F.\mathtt{qet}[\mathtt{stm}]\{F\} +_{[\![\mathtt{b}]\!]} f\right)$$
$$\mathtt{qet}[\mathtt{q} *= U]\{f\} \triangleq \lambda(s,\rho).f(s,\Phi_{U_{\mathtt{q}}})$$
$$\mathtt{qet}[\mathtt{x} := \mathtt{meas\ q}]\{f\} \triangleq \lambda(s,\rho).f(s[\mathtt{x} := 0],m_0) +_{p_0} f(s[\mathtt{x} := 1],m_1)$$

$$M_0 \triangleq \left(\begin{smallmatrix}1 & 0\\ 0 & 0\end{smallmatrix}\right) \qquad M_1 \triangleq \left(\begin{smallmatrix}0 & 0\\ 0 & 1\end{smallmatrix}\right)$$
$$x +_b y \triangleq bx + (1-b)y, \qquad \Phi_M \triangleq M\rho M^\dagger$$
$$p_k \triangleq tr(\Phi_{M_k}\rho) \qquad m_k \triangleq \frac{\Phi_{M_k}\rho}{p_k},\ \text{if } p_k \neq 0.$$

# Qet example for Cointoss (1/2)

The loop body:

```
x := true;
i := 0;
while x do {
    i := i + 1;
    q *= H;
    x := meas q
}
```

$\texttt{qet}[\texttt{i} := \texttt{i}{+}1; \texttt{q}{*}= \texttt{H}; \texttt{x} := \texttt{meas q}]\{g\}\,\langle s, \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}\rangle$

$\texttt{qet} \in \texttt{Prog} \to (\texttt{State} \to \mathbb{R}^{+\infty}) \to (\texttt{State} \to \mathbb{R}^{+\infty})$

# Qet example for Cointoss (1/2)

The loop body:

```
x := true;
i := 0;
while x do {
    i := i + 1;
    q *= H;
    x := meas q
}
```

$$\texttt{qet}[\texttt{i} := \texttt{i}{+}1; \texttt{q}{*}= \texttt{H}; \texttt{x} := \texttt{meas q}]\{g\}\,\langle s, \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}\rangle$$

$$= \texttt{qet}[\texttt{i} := \texttt{i}{+}1]\{\texttt{qet}[\texttt{q}{*}= \texttt{H}]\{\texttt{qet}[\texttt{x} := \texttt{meas q}]\{g\}\}\}\,\langle s, \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}\rangle$$

$$\texttt{qet} \in \texttt{Prog} \to (\texttt{State} \to \mathbb{R}^{+\infty}) \to (\texttt{State} \to \mathbb{R}^{+\infty})$$

# Qet example for Cointoss (1/2)

The loop body:

```
x := true;
i := 0;
while x do {
    i := i + 1;
    q *= H;
    x := meas q
}
```

$$\texttt{qet}[\texttt{i} := \texttt{i}{+}1; \texttt{q}{*}= \texttt{H}; \texttt{x} := \texttt{meas q}]\{g\}\,\langle s, \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}\rangle$$

$$= \texttt{qet}[\texttt{i} := \texttt{i}{+}1]\{\texttt{qet}[\texttt{q}{*}= \texttt{H}]\{\texttt{qet}[\texttt{x} := \texttt{meas q}]\{g\}\}\}\,\langle s, \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}\rangle$$

$$= \texttt{qet}[\texttt{i} := \texttt{i}{+}1]\{\texttt{qet}[\texttt{q}{*}= \texttt{H}]\{g[\texttt{x} := 0; m_0] +_{p_0} g[\texttt{x} := 1; m_1]\}\}\,\langle s, \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}\rangle$$

$$\texttt{qet} \in \texttt{Prog} \to (\texttt{State} \to \mathbb{R}^{+\infty}) \to (\texttt{State} \to \mathbb{R}^{+\infty})$$

# Qet example for Cointoss (1/2)

The loop body:

```
x := true;
i := 0;
while x do {
    i := i + 1;
    q *= H;
    x := meas q
}
```

$$\texttt{qet}[\texttt{i} := \texttt{i}{+}1; \texttt{q}{*}= \texttt{H}; \texttt{x} := \texttt{meas q}]\{g\}\,\langle s, \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}\rangle$$

$$= \texttt{qet}[\texttt{i} := \texttt{i}{+}1]\{\texttt{qet}[\texttt{q}{*}= \texttt{H}]\{\texttt{qet}[\texttt{x} := \texttt{meas q}]\{g\}\}\}\,\langle s, \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}\rangle$$

$$= \texttt{qet}[\texttt{i} := \texttt{i}{+}1]\{\texttt{qet}[\texttt{q}{*}= \texttt{H}]\{g[\texttt{x} := 0; m_0] +_{p_0} g[\texttt{x} := 1; m_1]\}\}\,\langle s, \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}\rangle$$

$$= \texttt{qet}[\texttt{i} := \texttt{i}{+}1]\{g[\texttt{x} := 0; m_0 \circ \Phi_H] +_{p_0 \circ \Phi_H} g[\texttt{x} := 1; m_1 \circ \Phi_H]\}\,\langle s, \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}\rangle$$

$$\texttt{qet} \in \texttt{Prog} \to (\texttt{State} \to \mathbb{R}^{+\infty}) \to (\texttt{State} \to \mathbb{R}^{+\infty})$$

# Qet example for Cointoss (1/2)

The loop body:

```
x := true;
i := 0;
while x do {
    i := i + 1;
    q *= H;
    x := meas q
}
```

$$\texttt{qet}[\texttt{i} := \texttt{i}{+}1; \texttt{q}{*}= \texttt{H}; \texttt{x} := \texttt{meas q}]\{g\}\,\langle s, \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}\rangle$$

$$= \texttt{qet}[\texttt{i} := \texttt{i}{+}1]\{\texttt{qet}[\texttt{q}{*}= \texttt{H}]\{\texttt{qet}[\texttt{x} := \texttt{meas q}]\{g\}\}\}\,\langle s, \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}\rangle$$

$$= \texttt{qet}[\texttt{i} := \texttt{i}{+}1]\{\texttt{qet}[\texttt{q}{*}= \texttt{H}]\{g[\texttt{x} := 0; m_0] +_{p_0} g[\texttt{x} := 1; m_1]\}\}\,\langle s, \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}\rangle$$

$$= \texttt{qet}[\texttt{i} := \texttt{i}{+}1]\{g[\texttt{x} := 0; m_0 \circ \Phi_H] +_{p_0 \circ \Phi_H} g[\texttt{x} := 1; m_1 \circ \Phi_H]\}\,\langle s, \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}\rangle$$

$$= g[\texttt{x} := 0, \texttt{i} := \texttt{i}{+}1; m_0 \circ \Phi_H] +_{p_0 \circ \Phi_H} g[\texttt{x} := 1, \texttt{i} := \texttt{i}{+}1; m_1 \circ \Phi_H]\langle s, \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}\rangle$$

$$\texttt{qet} \in \texttt{Prog} \to (\texttt{State} \to \mathbb{R}^{+\infty}) \to (\texttt{State} \to \mathbb{R}^{+\infty})$$

# Qet example for Cointoss (1/2)

The loop body:

```
x := true;
i := 0;
while x do {
    i := i + 1;
    q *= H;
    x := meas q
}
```

$$\texttt{qet}[\texttt{i} := \texttt{i}{+}1; \texttt{q}{*}= \texttt{H}; \texttt{x} := \texttt{meas q}]\{g\}\,\langle s, \left(\begin{smallmatrix}\alpha & \beta\\ \gamma & \delta\end{smallmatrix}\right)\rangle$$

$$= \texttt{qet}[\texttt{i} := \texttt{i}{+}1]\{\texttt{qet}[\texttt{q}{*}= \texttt{H}]\{\texttt{qet}[\texttt{x} := \texttt{meas q}]\{g\}\}\}\,\langle s, \left(\begin{smallmatrix}\alpha & \beta\\ \gamma & \delta\end{smallmatrix}\right)\rangle$$

$$= \texttt{qet}[\texttt{i} := \texttt{i}{+}1]\{\texttt{qet}[\texttt{q}{*}= \texttt{H}]\{g[\texttt{x} := 0; m_0] +_{p_0} g[\texttt{x} := 1; m_1]\}\}\,\langle s, \left(\begin{smallmatrix}\alpha & \beta\\ \gamma & \delta\end{smallmatrix}\right)\rangle$$

$$= \texttt{qet}[\texttt{i} := \texttt{i}{+}1]\{g[\texttt{x} := 0; m_0 \circ \Phi_H] +_{p_0 \circ \Phi_H} g[\texttt{x} := 1; m_1 \circ \Phi_H]\}\,\langle s, \left(\begin{smallmatrix}\alpha & \beta\\ \gamma & \delta\end{smallmatrix}\right)\rangle$$

$$= g[\texttt{x} := 0, \texttt{i} := \texttt{i}{+}1; m_0 \circ \Phi_H] +_{p_0 \circ \Phi_H} g[\texttt{x} := 1, \texttt{i} := \texttt{i}{+}1; m_1 \circ \Phi_H]\langle s, \left(\begin{smallmatrix}\alpha & \beta\\ \gamma & \delta\end{smallmatrix}\right)\rangle$$

$$\dots$$

$$= \frac{1+\beta+\gamma}{2} \cdot g\langle s^{\texttt{x}:=0}_{\texttt{i}:=\texttt{i}+1}, \left(\begin{smallmatrix}1 & 0\\ 0 & 0\end{smallmatrix}\right)\rangle + \frac{1-\beta-\gamma}{2} \cdot g\langle s^{\texttt{x}:=1}_{\texttt{i}:=\texttt{i}+1}, \left(\begin{smallmatrix}0 & 0\\ 0 & 1\end{smallmatrix}\right)\rangle$$

$$\texttt{qet} \in \texttt{Prog} \to (\texttt{State} \to \mathbb{R}^{+\infty}) \to (\texttt{State} \to \mathbb{R}^{+\infty})$$

# Qet example for Cointoss (2/2)

The loop:

```
x := true;
i := 0;
while x do {
    stm
}
```

$$\texttt{qet}[\texttt{while x do stm}]\{f\} = \mathsf{lfp}\left(\lambda g.\texttt{qet}[\texttt{stm}]\{g\} +_{[\![\texttt{x}]\!]} f\right)$$

with $\texttt{qet}[\texttt{stm}]\{g\} = \lambda\langle s, \left(\begin{smallmatrix}\alpha & \beta\\ \gamma & \delta\end{smallmatrix}\right)\rangle. \sum_{k\in\{0,1\}} \frac{1+(-1)^k(\beta+\gamma)}{2} \cdot g\langle s^{\texttt{x}:=k}_{\texttt{i}:=\texttt{i}+1}, \left(\begin{smallmatrix}1-k & 0\\ 0 & k\end{smallmatrix}\right)\rangle$

$$\texttt{qet} \in \texttt{Prog} \to (\texttt{State} \to \mathbb{R}^{+\infty}) \to (\texttt{State} \to \mathbb{R}^{+\infty})$$

# Qet example for Cointoss (2/2)

The loop:

```
x := true;
i := 0;
while x do {
    stm
}
```

$$\texttt{qet}[\texttt{while x do stm}]\{f\} = \mathsf{lfp}\left(\lambda g.\texttt{qet}[\texttt{stm}]\{g\} +_{[\![\texttt{x}]\!]} f\right)$$

with $\texttt{qet}[\texttt{stm}]\{g\} = \lambda\langle s, \left(\begin{smallmatrix}\alpha & \beta\\ \gamma & \delta\end{smallmatrix}\right)\rangle. \sum_{k\in\{0,1\}} \frac{1+(-1)^k(\beta+\gamma)}{2} \cdot g\langle s^{\texttt{x}:=k}_{\texttt{i}:=\texttt{i}+1}, \left(\begin{smallmatrix}1-k & 0\\ 0 & k\end{smallmatrix}\right)\rangle$

For $f \triangleq \lambda\langle s, \rho\rangle.s(\texttt{i})$, it holds that:

$$\texttt{qet}[\texttt{while x do stm}]\{f\} = \lambda\langle s, \left(\begin{smallmatrix}\alpha & \beta\\ \gamma & \delta\end{smallmatrix}\right)\rangle.s(\texttt{i}) + 2 - (\beta+\gamma)$$

$$\texttt{qet} \in \texttt{Prog} \to (\texttt{State} \to \mathbb{R}^{+\infty}) \to (\texttt{State} \to \mathbb{R}^{+\infty})$$

# Qet example for Cointoss (2/2)

The loop:

```
x := true;
i := 0;
while x do {
    stm
}
```

$$\texttt{qet}[\texttt{while x do stm}]\{f\} = \mathsf{lfp}\left(\lambda g.\texttt{qet}[\texttt{stm}]\{g\} +_{[\![\texttt{x}]\!]} f\right)$$

with $\texttt{qet}[\texttt{stm}]\{g\} = \lambda\langle s, \left(\begin{smallmatrix}\alpha & \beta\\ \gamma & \delta\end{smallmatrix}\right)\rangle. \sum_{k\in\{0,1\}} \frac{1+(-1)^k(\beta+\gamma)}{2} \cdot g\langle s^{\texttt{x}:=k}_{\texttt{i}:=\texttt{i}+1}, \left(\begin{smallmatrix}1-k & 0\\ 0 & k\end{smallmatrix}\right)\rangle$

For $f \triangleq \lambda\langle s, \rho\rangle.s(\texttt{i})$, it holds that:

$$\texttt{qet}[\texttt{while x do stm}]\{f\} = \lambda\langle s, \left(\begin{smallmatrix}\alpha & \beta\\ \gamma & \delta\end{smallmatrix}\right)\rangle.s(\texttt{i}) + 2 - (\beta+\gamma)$$

The expected value of i for Cointoss is:

$$\texttt{qet}[\texttt{while x do stm}]\{f\}\,[x := \texttt{true}, i := 0]\langle s, \left(\begin{smallmatrix}\alpha & \beta\\ \gamma & \delta\end{smallmatrix}\right)\rangle = 2 - (\beta+\gamma)$$

$$\texttt{qet} \in \texttt{Prog} \to (\texttt{State} \to \mathbb{R}^{+\infty}) \to (\texttt{State} \to \mathbb{R}^{+\infty})$$

# Adequacy

**Theorem [Adequacy]**

$\texttt{qet}[\texttt{stm}]\{f\}$ = the expected value computed by `stm` wrt the expectation $f$.

**Corollary**

*Computing* $\texttt{qet}[\texttt{stm}]\{f\}$ *is* $\Pi_2^0$*-complete.*

- It corresponds to $\mathrm{UTEST}_=$
- The undecidability comes from the *lfp*

# Approximation and restriction

## Approximation: upper invariance

$$(\llbracket \neg \mathtt{b} \rrbracket \cdot f \leq g \ \wedge \ \llbracket \mathtt{b} \rrbracket \cdot \mathtt{qet[stm]}\{g\} \leq g) \quad \Rightarrow \quad \mathtt{qet[while\ b\ do\ stm]}\{f\} \leq g$$

$\rightarrow$ it avoids computing fixpoints

## Restriction of expectations to polynomials over $\overline{\mathbb{Q}}$

$$\lambda(\{X_i\}_{1 \leq i \leq n}, (A_{j,k} + iB_{j,k})_{1 \leq j,k \leq 2^m}).\ P$$

with $P \in \overline{\mathbb{Q}}[X_1, \ldots, X_n, A_{1,1}, \ldots, A_{2^m,2^m}, B_{1,1}, \ldots, B_{2^m,2^m}]$, a polynomial of degree $\leq d$.

$\rightarrow$ Clifford+T does not escape the algebraic fragment!

# A decidability result

## Definition

Let *QINFER* be the problem of deciding whether

- some given polynomial expectations of degree $d$ over the algebraic numbers
- are solution of the inequalities obtained using upper invariance approximation.

## Theorem

*QINFER* can be decided in time doubly exponential in the size of the program.

$\rightarrow$ based on quantifier elimination.

# Conclusion

- Statically analyzing programs is difficult
- The situation is even worse for quantum programs
- We have shown completeness results in the arithmetical hierarchy for:
  - standard problems (AST, expectation, ...)
  - over a standard imperative quantum PL (Clifford+T)
- Obtained a decidability result under some reasonable restrictions:
  - upper bounds on expectations
  - polynomial functions of bounded degree