

# On the Hardness of Analyzing Quantum Programs Quantitatively

(published and presented at ESOP 2024)

Martin Avanzini<sup>‡</sup>, Georg Moser\*, **Romain Péchoux**<sup>†</sup> and Simon Perdrix<sup>†</sup>

<sup>‡</sup> Inria team Focus - Inria Côte d'Azur

\* University of Innsbruck

<sup>†</sup> Inria team Mocqua - CNRS, Inria, Université de Lorraine - LORIA

## QPL 24

July 17th, 2024



# (Classical) 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...



# 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 (imposes linearity)
  - ▶ *quantum interference* (amplitudes may cancel out)
  - ▶ *entanglement* (difficulty to approximate)
- ▶ quantum programs work on Hilbert spaces of **complex numbers**:
  - ▶ *continuous* domains (includes *non-computable* numbers)
  - ▶ where basic properties are undecidable (e.g., equality is not decidable)

This last problem is often underestimated!

# Quantum programs in this talk

Hybrid quantum-classical typed programs:

Integer  $\ni i ::= x \mid n \in \mathbb{N} \mid i + i \mid i \times i \mid \dots$   
Boolean  $\ni b ::= x \mid \text{true} \mid \text{false} \mid i = i \mid i < i \mid \neg b \mid b \wedge b \mid \dots$   
Expression  $\ni e ::= i \mid b$   
Prog  $\ni \text{stm} ::= \text{skip} \mid x := e \mid \text{stm}; \text{stm} \mid \text{if } b \text{ then } \text{stm} \text{ else } \text{stm}$   
 $\mid \text{while } b \text{ do } \text{stm} \mid q * = U \mid x := \text{meas } q$

Semantics defined over State  $\triangleq \{(s, \rho) \mid s \in \text{Store}, \rho \in \mathcal{D}(\mathbb{C}^{2^n})\}$

## Simple illustrative example: Cointoss(q)

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

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

## 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>)

## Simple illustrative example: Cointoss(q)

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

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

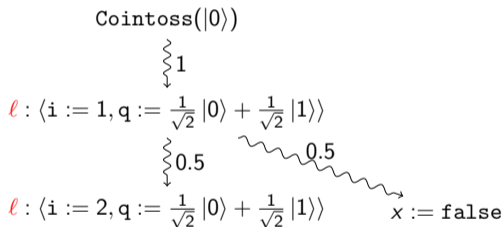
Cointoss( $|0\rangle$ )

$$\ell: \langle i := 1, q := \frac{1}{\sqrt{2}} |0\rangle + \frac{1}{\sqrt{2}} |1\rangle \rangle$$

## Simple illustrative example: Cointoss(q)

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

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





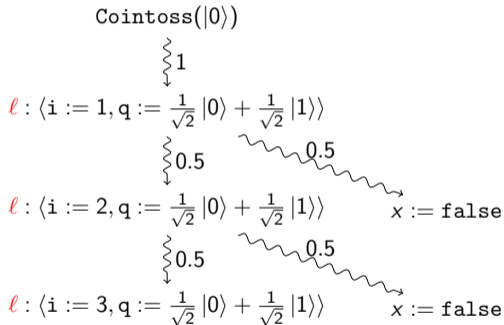
## Simple illustrative example: Cointoss(q)

```

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

```

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



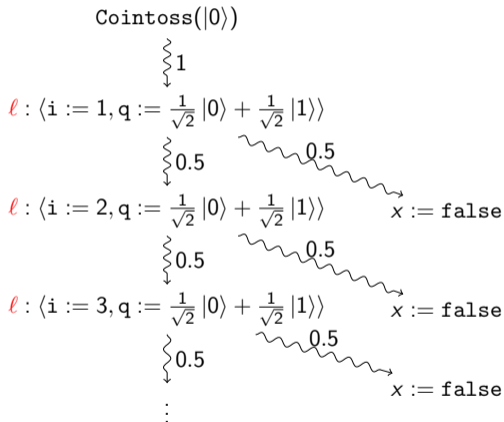
## Simple illustrative example: Cointoss(q)

```

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

```

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



## Quantitative properties of interest

- ▶ **AST : Almost-Sure Termination**
  - ▶ Termination with probability 1
- ▶ **PAST : Positive Almost-Sure Termination**
  - ▶ AST and finite mean derivation length
- ▶ **TEST<sub>=</sub> : Test the Expected value** of a given variable
  - ▶ Computed wrt a given *pre-expectation*  $f \in \text{State} \rightarrow \mathbb{R}^+$
  - ▶ Find **upper and lower-bounds** on the expected value, e.g.,  $\text{TEST}_{<}$ ,  $\text{TEST}_{\geq}$
  - ▶ Test the **finiteness** of the expected value:  $\text{TEST}_{\neq\infty}$
- ▶ All the above problems can be extended to their **Universal problem**
  - ▶ universal quantification over the input state, e.g.,  $\text{UPAST}$ ,  $\text{UTEST}_{=}$

## Example: Cointoss on input state $q \triangleq \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$

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

- ▶ Cointoss  $\in$  UAST (only the 1st iteration differs):

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

Take the pre-expectation  $f \in \text{State} \rightarrow \mathbb{R}^+$   
 $f \triangleq (s, \rho) \mapsto s(i)$

- ▶  $(\text{Cointoss}, f, \langle s, \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \rangle) \mapsto 2 - (\alpha\bar{\beta} + \bar{\alpha}\beta) \in \text{UTEST}_=$
- ▶ It also implies
  - ▶  $(\text{Cointoss}, f) \in \text{UTEST}_{\neq\infty}$
  - ▶ Cointoss  $\in$  UPAST

## Two restrictions

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

$$I \triangleq \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, X \triangleq \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Y \triangleq \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, Z \triangleq \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, H \triangleq \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix},$$

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

- ▶ 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 $\{f \mid f : \text{State} \rightarrow \mathbb{R}^+ \cap \overline{\mathbb{Q}}, f \text{ computable}\}$

## Reminder on the arithmetical hierarchy

### Definition [Levels of the arithmetical hierarchy]

$$\Pi_0^0 = \Sigma_0^0 \triangleq \text{REC} \quad (\text{decidable problems/recursive sets})$$

$$\Pi_{n+1}^0 \triangleq \{\psi \mid \exists \phi \in \Sigma_n^0, \forall \bar{x}. (\psi(\bar{x}) \iff \forall \bar{y}. \phi(\bar{x}, \bar{y}))\}$$

$$\Sigma_{n+1}^0 \triangleq \{\psi \mid \exists \phi \in \Pi_n^0, \forall \bar{x}. (\psi(\bar{x}) \iff \exists \bar{y}. \phi(\bar{x}, \bar{y}))\}$$

Examples of well-known (classical) complete problems:

$$\text{Halt}(p, d) \iff \exists t, p(d) \downarrow^t \text{ is } \Sigma_1^0 \text{ - complete}$$

$$\text{CoHalt}(p, d) \iff \neg \exists t, p(d) \downarrow^t \text{ is } \Pi_1^0 \text{ - complete}$$

$$\text{UHalt}(p) \iff \forall d, \exists t, p(d) \downarrow^t \text{ is } \Pi_2^0 \text{ - complete}$$

$$\text{UCoHalt}(p) \iff \forall d, \nexists t, p(d) \downarrow^t \text{ is } \Pi_1^0 \text{ - complete}$$

$$\text{UPolyHalt}(p) \iff \exists Q \in \mathbb{N}[X], \forall d, p(d) \downarrow^{Q(|d|)} \text{ is } \Sigma_2^0 \text{ - complete}$$

## Completeness results over Clifford+T

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

(\*): results not treated in the probabilistic case. Other results are surprising extensions of [Schnabl et al.2011] and [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.

$|term_p(d)| \triangleq \sup_{n \rightarrow \infty} |term_p^{\leq n}(d)|$ .

$$(p, d) \in \text{AST} \iff |term_p(d)| = 1$$

$$\iff \forall \epsilon \in \overline{\mathbb{Q}} \cap \mathbb{R}^+, \exists n \in \mathbb{N}, \underbrace{|term_p^{\leq n}(d)| \geq 1 - \epsilon}_{\Pi_2^0 \text{ within the Clifford+T fragment}}$$

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]
  - ▶  $\text{pt} \in \text{Prog} \rightarrow (\text{Store} \rightarrow \{0, 1\}) \rightarrow (\text{Store} \rightarrow \{0, 1\})$
- ▶ *Expectation transformers* for probabilistic programs [McIver-Morgan05]
  - ▶  $\text{wp} \in \text{Prog} \rightarrow (\text{Store} \rightarrow \mathbb{R}^{+\infty}) \rightarrow (\text{Store} \rightarrow \mathbb{R}^{+\infty})$
- ▶ *Quantum expectation transformers* for quantum programs [Avanzini et al. 22]
  - ▶  $\text{qet} \in \text{Prog} \rightarrow (\text{State} \rightarrow \mathbb{R}^{+\infty}) \rightarrow (\text{State} \rightarrow \mathbb{R}^{+\infty})$

## Quantum expectation transformers

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

$$\text{qet}[x := e]\{f\} \triangleq \lambda(s, \rho).f(s[x := \llbracket e \rrbracket], \rho)$$

$$\text{qet}[\text{stm}_1; \text{stm}_2]\{f\} \triangleq \text{qet}[\text{stm}_1]\{\text{qet}[\text{stm}_2]\{f\}\}$$

$$\text{qet}[\text{while } b \text{ do } \text{stm}]\{f\} \triangleq \text{lfp}(\lambda F.\text{qet}[\text{stm}]\{F\} + \llbracket b \rrbracket f)$$

$$\text{qet}[q^* = U]\{f\} \triangleq \lambda(s, \rho).f(s, \Phi_{U_q})$$

$$\text{qet}[x := \text{meas } q]\{f\} \triangleq \lambda(s, \rho).f(s[x := 0], m_0) + p_0 f(s[x := 1], m_1)$$

$$M_0 \triangleq \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

$$x +_b y \triangleq bx + (1 - b)y,$$

$$p_k \triangleq \text{tr}(\Phi_{M_k} \rho)$$

$$M_1 \triangleq \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\Phi_M \triangleq M \rho M^\dagger$$

$$m_k \triangleq \frac{\Phi_{M_k} \rho}{p_k}, \text{ if } p_k \neq 0.$$

## Qet example for Cointoss (1/2)

The loop body:

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

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

$\text{qet} \in \text{Prog} \rightarrow (\text{State} \rightarrow \mathbb{R}^{+\infty}) \rightarrow (\text{State} \rightarrow \mathbb{R}^{+\infty})$

## Qet example for Cointoss (1/2)

The loop body:

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

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

$$\text{qet} \in \text{Prog} \rightarrow (\text{State} \rightarrow \mathbb{R}^{+\infty}) \rightarrow (\text{State} \rightarrow \mathbb{R}^{+\infty})$$

## Qet example for Cointoss (1/2)

The loop body:

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

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

$$\text{qet} \in \text{Prog} \rightarrow (\text{State} \rightarrow \mathbb{R}^{+\infty}) \rightarrow (\text{State} \rightarrow \mathbb{R}^{+\infty})$$

## Qet example for Cointoss (1/2)

The loop body:

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

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

$$\text{qet} \in \text{Prog} \rightarrow (\text{State} \rightarrow \mathbb{R}^{+\infty}) \rightarrow (\text{State} \rightarrow \mathbb{R}^{+\infty})$$

## Qet example for Cointoss (1/2)

The loop body:

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

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

$$\text{qet} \in \text{Prog} \rightarrow (\text{State} \rightarrow \mathbb{R}^{+\infty}) \rightarrow (\text{State} \rightarrow \mathbb{R}^{+\infty})$$

## Qet example for Cointoss (1/2)

The loop body:

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

$$\begin{aligned}
 & \text{qet}[i := i+1; q* = H; x := \text{meas } q]\{g\} \langle s, \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \rangle \\
 &= \text{qet}[i := i+1]\{\text{qet}[q* = H]\{\text{qet}[x := \text{meas } q]\{g\}\}\} \langle s, \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \rangle \\
 &= \text{qet}[i := i+1]\{\text{qet}[q* = H]\{g[x := 0; m_0] + p_0 g[x := 1; m_1]\}\} \langle s, \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \rangle \\
 &= \text{qet}[i := i+1]\{g[x := 0; m_0 \circ \Phi_H] + p_0 \circ \Phi_H g[x := 1; m_1 \circ \Phi_H]\} \langle s, \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \rangle \\
 &= g[x := 0, i := i+1; m_0 \circ \Phi_H] + p_0 \circ \Phi_H g[x := 1, i := i+1; m_1 \circ \Phi_H] \langle s, \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \rangle \\
 &\dots \\
 &= \frac{1 + \beta + \gamma}{2} \cdot g \langle s_{i:=i+1}^{x:=0}, \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \rangle + \frac{1 - \beta - \gamma}{2} \cdot g \langle s_{i:=i+1}^{x:=1}, \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \rangle
 \end{aligned}$$

$$\text{qet} \in \text{Prog} \rightarrow (\text{State} \rightarrow \mathbb{R}^{+\infty}) \rightarrow (\text{State} \rightarrow \mathbb{R}^{+\infty})$$



## Qet example for Cointoss (2/2)

The loop:

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

$$\text{qet}[\text{while } x \text{ do } \text{stm}]\{f\} = \text{lfp}(\lambda g. \text{qet}[\text{stm}]\{g\} + \llbracket x \rrbracket f)$$
$$\text{with } \text{qet}[\text{stm}]\{g\} = \lambda \langle s, \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \rangle. \sum_{k \in \{0,1\}} \frac{1 + (-1)^k(\beta + \gamma)}{2} \cdot g \langle s_{i:=i+1}^{x:=k}, \begin{pmatrix} 1-k & 0 \\ 0 & k \end{pmatrix} \rangle$$

$$\text{qet} \in \text{Prog} \rightarrow (\text{State} \rightarrow \mathbb{R}^{+\infty}) \rightarrow (\text{State} \rightarrow \mathbb{R}^{+\infty})$$

## Qet example for Cointoss (2/2)

The loop:

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

$$\text{qet}[\text{while } x \text{ do } \text{stm}]\{f\} = \text{lfp}(\lambda g. \text{qet}[\text{stm}]\{g\} + \llbracket x \rrbracket f)$$

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

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

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

$$\text{qet} \in \text{Prog} \rightarrow (\text{State} \rightarrow \mathbb{R}^{+\infty}) \rightarrow (\text{State} \rightarrow \mathbb{R}^{+\infty})$$

## Qet example for Cointoss (2/2)

The loop:

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

$$\text{qet}[\text{while } x \text{ do } \text{stm}]\{f\} = \text{lfp}(\lambda g. \text{qet}[\text{stm}]\{g\} + \llbracket x \rrbracket f)$$

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

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

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

The expected value of  $i$  for Cointoss is:

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

$$\text{qet} \in \text{Prog} \rightarrow (\text{State} \rightarrow \mathbb{R}^{+\infty}) \rightarrow (\text{State} \rightarrow \mathbb{R}^{+\infty})$$

# Adequacy

## Theorem [Adequacy]

$\text{qet}[\text{stm}]\{f\}$  is exactly the expected value computed by  $\text{stm}$  wrt the expectation  $f$ .

## Corollary

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

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

## Approximation and restriction

Approximation: upper invariance

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

→ it avoids computing fixpoints

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

→ 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.

→ based on quantifier elimination.

# Conclusion

- ▶ Statically analyzing programs is difficult and 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:
  - ▶ Clifford+T
  - ▶ upper bounds on expectations
  - ▶ polynomial functions of bounded degree
- ▶ WIP with Avanzini, Diaz-Caro and Hainry: extension to functional programs