Adding Measurement to van Tonder's Calculus

Computer Science Undegraduate Thesis

Alejandro Díaz-Caro
Universidad Nacional de Rosario - Argentina

Advisor
Dr. Manuel Gadella
Universidad de Valladolid - Spain

Co-Advisor
Dr. Pablo Martínez Lopez
Universidad Nacional de La Plata - Argentina

Outline

A brief memory refresher

- van Tonder's calculus provides quantum concepts to the λ-calculus

A brief memory refresher

- van Tonder's calculus provides quantum concepts to the λ-calculus
- The syntax is the following:

Syntax of λ_q

$t ::=$		term
	x	variable
	$(\lambda x.t)$	abstraction
	$(t\ t)$	aplication
	c	constant
	$!t$	nonlinear term
	$(\lambda !x.t)$	nonlinear abstraction
$c ::=$		
	$0 \mid 1 \mid H \mid cnot \mid X \mid \ldots$	constants

A brief memory refresher

- van Tonder's calculus provides quantum concepts to the λ-calculus
- The syntax is the following:

Syntax of λ_q

$t ::=$		term
	x	variable
	$(\lambda x.t)$	abstraction
	$(t\ t)$	aplication
	c	constant
	$!t$	nonlinear term
	$(\lambda !x.t)$	nonlinear abstraction
$c ::=$		
	$0 \mid 1 \mid H \mid cnot \mid X \mid \ldots$	constants

- Nonlinear terms are used to distinguish definite terms from not-definite terms

A brief memory refresher (II)

Definition

Let us call a subexpresion definite with respect to a computational basis if it is textually the same in all branches of the superposition.

A brief memory refresher (II)

Definition

Let us call a subexpresion definite with respect to a computational basis if it is textually the same in all branches of the superposition.

Example

$$\frac{1}{\sqrt{2}}(|(\lambda x.0)\ 0\rangle + |(\lambda x.0)\ 1\rangle)$$

- the subexpresion $(\lambda x.0)$ is definite
- the argument $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ is non-definite

A brief memory refresher (III)

"Rules of use"

- Nonlinear terms will be guaranteed to be definite with respect to the computational basis

A brief memory refresher (III)

"Rules of use"

- Nonlinear terms will be guaranteed to be definite with respect to the computational basis
- $(\lambda !x.t)$: denotes functions of nonlinear arguments
- $(\lambda x.t)$: denotes functions of linear arguments

A brief memory refresher (III)

"Rules of use"

- Nonlinear terms will be guaranteed to be definite with respect to the computational basis
- $(\lambda!x.t)$: denotes functions of nonlinear arguments
- $(\lambda x.t)$: denotes functions of linear arguments
- A linear abstraction may use a nonlinear argument any number of times in its body, or not at all, but a linear argument must appear exactly once in the function body

A brief memory refresher (III)

"Rules of use"

- Nonlinear terms will be guaranteed to be definite with respect to the computational basis
- $(\lambda !x.t)$: denotes functions of nonlinear arguments
- $(\lambda x.t)$: denotes functions of linear arguments
- A linear abstraction may use a nonlinear argument any number of times in its body, or not at all, but a linear argument must appear exactly once in the function body

All this rules are given formally in a set of well-formedness rules

A brief memory refresher (IV)

- To prevent terms of the form $!t$ from being evaluated, he extends the definition of values as follows:

Values in λ_q

v	$::=$		values:
		x	variable
		c	constant
		$(\lambda x.t)$	linear abstraction
		$(\lambda !x.t)$	nonlinear abstraction
		$!t$	!-suspension

A brief memory refresher (V)

- To make the process reversible, he adds a history track which saves the reduced terms in each step and the operation that has been applied

A brief memory refresher (V)

- To make the process reversible, he adds a history track which saves the reduced terms in each step and the operation that has been applied

Example of transition rule with history track

$$\frac{}{\mathcal{H}; ((\lambda x.t)\ v) \rightarrow \mathcal{H}; ((\lambda x.\overline{t}_x)\ \_); t[v/x]} \quad (\beta)$$

where $\overline{t}_x$ is obtained from t by recursively replacing all subterms that do not contain x with the placeholder simbol and keeping x

A brief memory refresher (V)

- To make the process reversible, he adds a history track which saves the reduced terms in each step and the operation that has been applied

Example of transition rule with history track

$$\frac{}{\mathcal{H}; ((\lambda x.t)\ v) \rightarrow \mathcal{H}; ((\lambda x.\overline{t}_x)\ \_); t[v/x]} \quad (\beta)$$

where $\overline{t}_x$ is obtained from t by recursively replacing all subterms that do not contain x with the placeholder simbol and keeping x

- For gates there are some specific rules

Example of Hadamard rule

$H\ 0$ reduces to $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$

Motivation

- André van Tonder's linear untyped λ-calculus does not include Measurement Operation

Motivation

- André van Tonder's linear untyped λ-calculus does not include Measurement Operation
- He declares that any algorithm can be redesigned to defer measurements

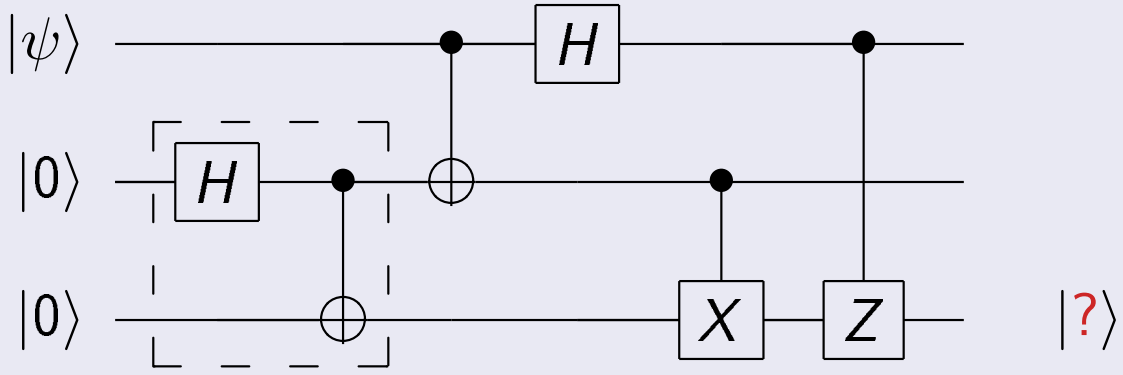
Motivation

- André van Tonder's linear untyped λ-calculus does not include Measurement Operation
- He declares that any algorithm can be redesigned to defer measurements
- But it can generate misunderstanding in some algorithms

Motivation

- André van Tonder's linear untyped λ-calculus does not include Measurement Operation
- He declares that any algorithm can be redesigned to defer measurements
- But it can generate misunderstanding in some algorithms

Example: Teleportation with deferred measurement



Here it is not clear that Alice and Bob can be away from each other

Motivation (II)

Same example in λ_q

$$\begin{aligned}\text{teleport } x \longrightarrow\ & \textbf{let } (e_1, e_2) = \text{epr } \textbf{in} \\ & \textbf{let } (x', y') = \text{alice } (x, e_1) \textbf{ in} \\ & \quad \text{bob } (x', y', e_2)\end{aligned}$$

where

$$\text{alice } (x, e_1) \longrightarrow \textbf{let } (x', y') = \mathit{cnot}\ (x, e_1) \textbf{ in } ((H\ x'), y')$$

$$\begin{aligned}\text{bob } (x', y', e_2) \longrightarrow\ & \textbf{let } (y'', e_2') = cX\ (y', e_2) \textbf{ in} \\ & \textbf{let } (x'', e_2'') = cZ\ (x', e_2') \textbf{ in} \\ & \quad (x'', y'', e_2'')\end{aligned}$$

$$\text{epr} \equiv \mathit{cnot}\ ((H\ 0), 0)$$

Motivation (II)

Same example in λ_q

$$\begin{aligned}\text{teleport } x \longrightarrow\ & \textbf{let } (e_1, e_2) = \text{epr } \textbf{in} \\ & \textbf{let } (x', y') = \text{alice } (x, e_1) \textbf{ in} \\ & \quad \text{bob } (x', y', e_2)\end{aligned}$$

where

$$\text{alice } (x, e_1) \longrightarrow \textbf{let } (x', y') = \mathit{cnot}\ (x, e_1) \textbf{ in } ((H\ x'), y')$$

$$\begin{aligned}\text{bob } (x', y', e_2) \longrightarrow\ & \textbf{let } (y'', e_2') = cX\ (y', e_2) \textbf{ in} \\ & \textbf{let } (x'', e_2'') = cZ\ (x', e_2') \textbf{ in} \\ & \quad (x'', y'', e_2'')\end{aligned}$$

$$\text{epr} \equiv \mathit{cnot}\ ((H\ 0), 0)$$

Motivation (II)

Same example in λ_q

$$\begin{aligned}\text{teleport } x \longrightarrow\ & \textbf{let } (e_1, e_2) = \text{epr } \textbf{in} \\ & \textbf{let } (x', y') = M_2 \text{ alice } (x, e_1) \textbf{ in} \\ & \quad \text{bob } (x', y', e_2)\end{aligned}$$

where

$$\text{alice } (x, e_1) \longrightarrow \textbf{let } (x', y') = \mathit{cnot}\ (x, e_1) \textbf{ in } ((H\ x'), y')$$

$$\begin{aligned}\text{bob } (x', y', e_2) \longrightarrow\ & \textbf{let } (y'', e_2') = cX\ (y', e_2) \textbf{ in} \\ & \textbf{let } (x'', e_2'') = cZ\ (x', e_2') \textbf{ in} \\ & \quad (x'', y'', e_2'')\end{aligned}$$

$$\text{epr} \equiv \mathit{cnot}\ ((H\ 0), 0)$$

Motivation (II)

Same example in λ_q

$$\begin{aligned}\text{teleport } x \longrightarrow\ & \textbf{let } (e_1, e_2) = \text{epr } \textbf{in} \\ & \textbf{let } (x', y') = M_2 \text{ alice } (x, e_1) \textbf{ in} \\ & \quad \text{bob } (x', y', e_2)\end{aligned}$$

where

$$\text{alice } (x, e_1) \longrightarrow \textbf{let } (x', y') = \mathit{cnot}\ (x, e_1) \textbf{ in } ((H\ x'), y')$$

$$\text{bob } (x', y', e_2) \longrightarrow$$

$$\text{epr} \equiv \mathit{cnot}\ ((H\ 0), 0)$$

Motivation (III)

Goal

The intention of this work is to add the measurement operation to the untyped λ_q, keeping it untyped

1^{st} step: Probabilistic Transition Rules

- Measurements are inherently probabilistic

1^{st} step: Probabilistic Transition Rules

- Measurements are inherently probabilistic
- We need probabilistic transition rules

1^{st} step: Probabilistic Transition Rules

- Measurements are inherently probabilistic
- We need probabilistic transition rules
- Following Di Pierro *et. al.*, we use a simple model in which each transition rule may have several conclusions, each with an associated happening probability

1^{st} step: Probabilistic Transition Rules

- Measurements are inherently probabilistic
- We need probabilistic transition rules
- Following Di Pierro *et. al.*, we use a simple model in which each transition rule may have several conclusions, each with an associated happening probability

Example

$$\frac{\textit{Premises over } P}{\begin{array}{l} P \rightarrow_p Q_1 \\ P \rightarrow_q Q_2 \end{array}}$$

P goes to Q_1 with probability p and to Q_2 with probability q

1^{st} step: Probabilistic Transition Rules

- Measurements are inherently probabilistic
- We need probabilistic transition rules
- Following Di Pierro *et. al.*, we use a simple model in which each transition rule may have several conclusions, each with an associated happening probability

Example

$$\frac{\textit{Premises over } P}{\begin{array}{l} P \rightarrow_p Q_1 \\ P \rightarrow_q Q_2 \end{array}}$$

P goes to Q_1 with probability p and to Q_2 with probability q

Problem

- To know the probability of making a transition to each state, we need to know the "shape" of the qubit

1^{st} step: Probabilistic Transition Rules

- Measurements are inherently probabilistic
- We need probabilistic transition rules
- Following Di Pierro *et. al.*, we use a simple model in which each transition rule may have several conclusions, each with an associated happening probability

Example

$$\frac{\textit{Premises over } P}{\begin{array}{l} P \rightarrow_p Q_1 \\ P \rightarrow_q Q_2 \end{array}}$$

P goes to Q_1 with probability p and to Q_2 with probability q

Problem

- To know the probability of making a transition to each state, we need to know the "shape" of the qubit
- The λ_q syntax has no restriction on the shape of the qubits, they are taken as constants

2^{nd} step: Providing Syntax for Qubits

- We need to split the constants into qubits, gates and measurements:

2^{nd} step: Providing Syntax for Qubits

- We need to split the constants into qubits, gates and measurements:

Syntax for λ_q^M

$t ::=$		*terms:*
	x	*variable*
	$(\lambda x.t)$	*abstraction*
	$(t\ t)$	*aplication*
	$!t$	*nonlinear term*
	$(\lambda !x.t)$	*nonlinear abstraction*
	c_U	*constant for gates*
	q	*constant for qubits*
	M_n	*constant for measurement*

2^{nd} step: Providing Syntax for Qubits (II)

- Measurements are represented by constants M_n, where n is the number of qubits to measure

2^{nd} step: Providing Syntax for Qubits (II)

- Measurements are represented by constants M_n, where n is the number of qubits to measure
- Gates are constants like

$$c_U ::= H \mid cnot \mid X \mid Z \mid q(q)^T + q(q)^T \mid \ldots$$

2^{nd} step: Providing Syntax for Qubits (II)

- Measurements are represented by constants M_n, where n is the number of qubits to measure
- Gates are constants like

$$c_U ::= H \mid cnot \mid X \mid Z \mid q(q)^T + q(q)^T \mid \ldots$$

- For qubits, we provide the following syntax

2^{nd} step: Providing Syntax for Qubits (II)

- Measurements are represented by constants M_n, where n is the number of qubits to measure
- Gates are constants like

$$c_U ::= H \mid cnot \mid X \mid Z \mid q(q)^T + q(q)^T \mid \ldots$$

- For qubits, we provide the following syntax

Syntax for qubits

$q ::=$		*qubits:*
	$\lvert 0\rangle \mid \lvert 1\rangle$	*definite qubits*
	$(q \otimes q)$	*tensorial product*
	$(q + q)$	*superposition*
	$\alpha(q)$	*scalar product*

2^{nd} step: Providing Syntax for Qubits (III)

Problem

- We need to restrict when a string is a qubit and when it is not. We need to add rules for well-formed terms

3^{rd} step: Rules for well-formed terms

Rules for well-formedness. We add the following rules to van Tonder's:

3^{rd} step: Rules for well-formed terms

Rules for well-formedness. We add the following rules to van Tonder's:

Rules for well-formedness (I)

$$\frac{n \in \mathbb{N}}{\vdash M_n} \qquad M$$

3^{rd} step: Rules for well-formed terms

Rules for well-formedness. We add the following rules to van Tonder's:

Rules for well-formedness (I)

$$\frac{n \in \mathbb{N}}{\vdash M_n} \quad M$$

$$\frac{}{\vdash c_U} \quad Gate$$

3^{rd} step: Rules for well-formed terms

Rules for well-formedness. We add the following rules to van Tonder's:

Rules for well-formedness (I)

$$\frac{n \in \mathbb{N}}{\vdash M_n} \; M$$

$$\frac{}{\vdash c_U} \; Gate$$

$$\frac{}{\vdash ! \, |0\rangle} \; Zero$$

3^{rd} step: Rules for well-formed terms

Rules for well-formedness. We add the following rules to van Tonder's:

Rules for well-formedness (I)

$$\frac{n \in \mathbb{N}}{\vdash M_n}\ M$$

$$\frac{}{\vdash c_U}\ Gate$$

$$\frac{}{\vdash ! \, |0\rangle}\ Zero$$

$$\frac{}{\vdash ! \, |1\rangle}\ One$$

3^{rd} step: Rules for well-formed terms

Rules for well-formedness. We add the following rules to van Tonder's:

Rules for well-formedness (I)

$$\frac{n \in \mathbb{N}}{\vdash M_n}\ M$$

$$\frac{}{\vdash c_U}\ Gate$$

$$\frac{}{\vdash ! \left|0\right\rangle}\ Zero$$

$$\frac{}{\vdash ! \left|1\right\rangle}\ One$$

$$\frac{\Gamma \vdash q_1 \qquad \Delta \vdash q_2}{\Gamma, \Delta \vdash q_1 \otimes q_2}\ Tensor$$

3^{rd} step: Rules for well-formed terms

Rules for well-formedness. We add the following rules to van Tonder's:

Rules for well-formedness (I)

$$\frac{n \in \mathbb{N}}{\vdash M_n}\ M$$

$$\frac{}{\vdash c_U}\ Gate$$

$$\frac{}{\vdash\, !\,|0\rangle}\ Zero$$

$$\frac{}{\vdash\, !\,|1\rangle}\ One$$

$$\frac{\Gamma \vdash q_1 \qquad \Delta \vdash q_2}{\Gamma, \Delta \vdash q_1 \otimes q_2}\ Tensor$$

$$\frac{\Gamma \vdash\, !\, q_1 \qquad \Delta \vdash\, !\, q_2}{\Gamma, \Delta \vdash\, !\, q_1 \otimes\, !\, q_2}\ !Tensor$$

3^{rd} step: Rules for well-formed terms (II)

Rules for well-formed terms (II)

$$\frac{\sum_{i=1}^{2^n} |\alpha_i|^2 = 1 \qquad \alpha_i \in \mathbb{C}, i = 1, \ldots, 2^n}{\vdash \alpha_0(!\,|0\rangle \otimes \cdots \otimes !\,|0\rangle) + \cdots + \alpha_{2^n}(!\,|1\rangle \otimes \cdots \otimes !\,|1\rangle)} \; \textit{Superpos}$$

3^{rd} step: Rules for well-formed terms (II)

Rules for well-formed terms (II)

$$\frac{\sum_{i=1}^{2^n} |\alpha_i|^2 = 1 \qquad \alpha_i \in \mathbb{C}, i = 1, \ldots, 2^n}{\vdash \alpha_0(!\,|0\rangle \otimes \cdots \otimes !\,|0\rangle) + \cdots + \alpha_{2^n}(!\,|1\rangle \otimes \cdots \otimes !\,|1\rangle)} \; \textit{Superpos}$$

$$\frac{\alpha_r = 0, r \in \{1, \ldots, 2^n\} \qquad \Gamma \vdash \sum_{i=1}^{2^n} \alpha_i q_i}{\Gamma \vdash \sum_{\substack{i=1 \\ i \neq r}}^{2^n} \alpha_i q_i} \; \textit{Simplif}$$

3^{rd} step: Rules for well-formed terms (II)

Rules for well-formed terms (II)

$$\frac{\sum_{i=1}^{2^n} |\alpha_i|^2 = 1 \qquad \alpha_i \in \mathbb{C}, i = 1, \ldots, 2^n}{\vdash \alpha_0(!\,|0\rangle \otimes \cdots \otimes !\,|0\rangle) + \cdots + \alpha_{2^n}(!\,|1\rangle \otimes \cdots \otimes !\,|1\rangle)} \; \textit{Superpos}$$

$$\frac{\alpha_r = 0, r \in \{1, \ldots, 2^n\} \qquad \Gamma \vdash \sum_{i=1}^{2^n} \alpha_i q_i}{\Gamma \vdash \sum_{\substack{i=1 \\ i \neq r}}^{2^n} \alpha_i q_i} \; \textit{Simplif}$$

- The syntax previously defined will be the syntax of "pre-terms"

3^{rd} step: Rules for well-formed terms (II)

Rules for well-formed terms (II)

$$\frac{\sum_{i=1}^{2^n} |\alpha_i|^2 = 1 \qquad \alpha_i \in \mathbb{C}, i = 1, \ldots, 2^n}{\vdash \alpha_0(!\,|0\rangle \otimes \cdots \otimes !\,|0\rangle) + \cdots + \alpha_{2^n}(!\,|1\rangle \otimes \cdots \otimes !\,|1\rangle)} \; \textit{Superpos}$$

$$\frac{\alpha_r = 0, r \in \{1, \ldots, 2^n\} \qquad \Gamma \vdash \sum_{i=1}^{2^n} \alpha_i q_i}{\Gamma \vdash \sum_{\substack{i=1 \\ i \neq r}}^{2^n} \alpha_i q_i} \; \textit{Simplif}$$

- The syntax previously defined will be the syntax of "pre-terms"
- A term is a pre-term $\ddot{t}$ if $\vdash \ddot{t}$ can be infered from the rules for well-formedness

Operational Semantic

Now, we can to add a transition rule for measurements

Transition rule for measurements

$$\text{M:}\ \frac{q = \sum_{i=1}^{2^n} \alpha_i q_i}{\mathcal{H}; (M_n\ q) \rightarrow_{|\alpha_i|^2}\ q_i}$$

Where

$$q_i = !q_{1i} \otimes !q_{2i} \otimes \cdots \otimes !q_{ni}$$

with

$$!q_{ki} = !\,|0\rangle \ \text{ or } !\,|1\rangle \quad \forall k = 1 \ldots n$$

This rule discards the history track because measurement operations are irreversible

A remark about confluence

A remark about confluence

- Let us consider the following expresion

$$(\lambda x.(x\ x))\ (M_1\ (\frac{1}{\sqrt{2}}!\,|0\rangle + \frac{1}{\sqrt{2}}!\,|1\rangle))$$

A remark about confluence

- Let us consider the following expresion

$$(\lambda x.(x\ x))\ (M_1\ (\frac{1}{\sqrt{2}}!\,|0\rangle + \frac{1}{\sqrt{2}}!\,|1\rangle))$$

It is not well-formed! (the linear argument appears twice in the body function)

A remark about confluence

- Let us consider the following expresion

$$(\lambda x.(x\ x))\ (M_1\ (\frac{1}{\sqrt{2}}!\,|0\rangle + \frac{1}{\sqrt{2}}!\,|1\rangle))$$

  It is not well-formed! (the linear argument appears twice in the body function)
- Abstraction must be nonlinear

$$(\lambda !x.(x\ x))\ (M_1\ (\frac{1}{\sqrt{2}}!\,|0\rangle + \frac{1}{\sqrt{2}}!\,|1\rangle))$$

A remark about confluence

- Let us consider the following expresion

$$(\lambda x.(x\ x))\ (M_1\ (\frac{1}{\sqrt{2}}!\,|0\rangle + \frac{1}{\sqrt{2}}!\,|1\rangle))$$

  It is not well-formed! (the linear argument appears twice in the body function)

- Abstraction must be nonlinear

$$(\lambda !x.(x\ x))\ (M_1\ (\frac{1}{\sqrt{2}}!\,|0\rangle + \frac{1}{\sqrt{2}}!\,|1\rangle))$$

  The argument is linear, so M_1 has to apply first, producing a nonlinear output ($!\,|0\rangle$ or $!\,|1\rangle$)

A remark about confluence

- Let us consider the following expresion

$$(\lambda x.(x\ x))\ (M_1\ (\frac{1}{\sqrt{2}}!\,|0\rangle + \frac{1}{\sqrt{2}}!\,|1\rangle))$$

  It is not well-formed! (the linear argument appears twice in the body function)
- Abstraction must be nonlinear

$$(\lambda !x.(x\ x))\ (M_1\ (\frac{1}{\sqrt{2}}!\,|0\rangle + \frac{1}{\sqrt{2}}!\,|1\rangle))$$

  The argument is linear, so M_1 has to apply first, producing a nonlinear output ($!\,|0\rangle$ or $!\,|1\rangle$)
- Let us consider a *promotion* of the argument

$$(\lambda !x.(x\ x))\ !(M_1\ (\frac{1}{\sqrt{2}}!\,|0\rangle + \frac{1}{\sqrt{2}}!\,|1\rangle))$$

A remark about confluence

- Let us consider the following expresion

$$(\lambda x.(x\ x))\ (M_1\ (\frac{1}{\sqrt{2}}!\,|0\rangle + \frac{1}{\sqrt{2}}!\,|1\rangle))$$

  It is not well-formed! (the linear argument appears twice in the body function)
- Abstraction must be nonlinear

$$(\lambda !x.(x\ x))\ (M_1\ (\frac{1}{\sqrt{2}}!\,|0\rangle + \frac{1}{\sqrt{2}}!\,|1\rangle))$$

  The argument is linear, so M_1 has to apply first, producing a nonlinear output ($!\,|0\rangle$ or $!\,|1\rangle$)
- Let us consider a *promotion* of the argument

$$(\lambda !x.(x\ x))\ !(M_1\ (\frac{1}{\sqrt{2}}!\,|0\rangle + \frac{1}{\sqrt{2}}!\,|1\rangle))$$

  In this case, it is allowed to copy the measurement twice, but this is the only reduction strategy because of the ! mark

Example: Teleportation algorithm

Teleportation in λ_q^M

$$\begin{array}{rl} \text{teleport } q \longrightarrow_1 & \textbf{let } x \otimes y = \text{epr } \textbf{in} \\ & \textbf{let } b_1 \otimes b_2 = M_2 \text{ alice } (q, x) \textbf{ in} \\ & \quad \text{bob } (b_1, b_2, y) \end{array}$$

where

$\text{epr} \equiv \mathit{cnot}\ ((H\ !\,|0\rangle \otimes !\,|0\rangle)$

$\text{alice } (q, x) \longrightarrow_1 \textbf{let } r \otimes w = \mathit{cnot}\ \ q \otimes x \textbf{ in}$

$((H\ r) \otimes w)$

$\text{bob } (b_1, b_2, y) \longrightarrow_1 (\text{zed } b_1)\ (\text{ex } b_2)\ y$

. . . . . .

$\text{ex } b_2 \longrightarrow_1 !\,|0\rangle\ {b_2}^T + !\,|1\rangle\ (X\ b_2)^T$

$\text{zed } b_1 \longrightarrow_1 Z(!\,|0\rangle\ (!\,|0\rangle)^T + b_1 (!\,|1\rangle)^T) - !\,|0\rangle\ (!\,|1\rangle)^T + (X\ b_1)(!\,|1\rangle)^T$

$(\text{ex } b_2)$ is just X^{b_2} and $(\text{zed } b_1)$ is Z^{b_1}

Conclusions

The four most important properties of quantum computing, reversibility, entanglement, no-clonning and measurement, are now in the calculus

Conclusions

The four most important properties of quantum computing, reversibility, entanglement, no-clonning and measurement, are now in the calculus

- To add reversibility, van Tonder added a history track

Conclusions

The four most important properties of quantum computing, reversibility, entanglement, no-clonning and measurement, are now in the calculus

- To add reversibility, van Tonder added a history track
- To avoid having entanglement between the history track and the computational registry, he used the concepts of linear algebra to distinguish when a term is definite and when it is not

Conclusions

The four most important properties of quantum computing, reversibility, entanglement, no-clonning and measurement, are now in the calculus

- To add reversibility, van Tonder added a history track
- To avoid having entanglement between the history track and the computational registry, he used the concepts of linear algebra to distinguish when a term is definite and when it is not
- To prevent cloning, he made rules for well-formed terms

Conclusions

The four most important properties of quantum computing, reversibility, entanglement, no-clonning and measurement, are now in the calculus

- To add reversibility, van Tonder added a history track
- To avoid having entanglement between the history track and the computational registry, he used the concepts of linear algebra to distinguish when a term is definite and when it is not
- To prevent cloning, he made rules for well-formed terms
- To add measurement, we used probabilistic transition rules and provided a more detailed syntax with rules for well-formed terms