Predicative Quantum Programming

Anya Tafliovich

University of Toronto

August 22, 2009

taking a step back

What are we trying to achieve?

taking a step back

What are we trying to achieve?

- build a quantum computer

taking a step back

What are we trying to achieve?

- build a quantum computer
- write programs for it

taking a step back

What are we trying to achieve?

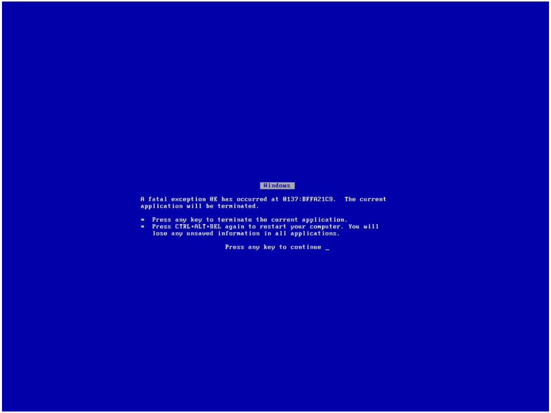
- build a quantum computer
- write programs for it

on writing programs

- Computer scientists have some experience with it.

on writing programs

- Computer scientists have some experience with it.



on writing programs

- Computer scientists have some experience with it.
- Ad hoc programming does not work. Things go wrong.

on writing programs

- Computer scientists have some experience with it.
- Ad hoc programming does not work. Things go wrong.
- Need a systematic approach to hardware and software development.

on writing programs

- Computer scientists have some experience with it.
- Ad hoc programming does not work. Things go wrong.
- Need a systematic approach to hardware and software development.
- Hardware and software correctness can be machine verifiable, given a good framework.

on writing programs

- Computer scientists have some experience with it.
- Ad hoc programming does not work. Things go wrong.
- Need a systematic approach to hardware and software development.
- Hardware and software correctness can be machine verifiable, given a good framework.

so, what's a refinement calculus?

so, what's a refinement calculus?

- start with specification: what we want

so, what's a refinement calculus?

- start with specification: what we want
- end with program: how we do it

so, what's a refinement calculus?

- start with specification: what we want
- end with program: how we do it
- move step by step from specification to program

so, what's a refinement calculus?

- start with specification: what we want
- end with program: how we do it
- move step by step from specification to program
- each step is justified by a law (and verified by a machine)

so, what's a refinement calculus?

- start with specification: what we want
- end with program: how we do it
- move step by step from specification to program
- each step is justified by a law (and verified by a machine)
- result: correct program

so, what's a refinement calculus?

- start with specification: what we want
- end with program: how we do it
- move step by step from specification to program
- each step is justified by a law (and verified by a machine)
- result: correct program
- also: time, space, probabilistic, etc. analysis

my research

A unified framework for

my research

A unified framework for

- writing specifications

my research

A unified framework for

- writing specifications
- developing programs

my research

A unified framework for

- writing specifications
- developing programs
- performing complexity analysis

my research

A unified framework for

- writing specifications
- developing programs
- performing complexity analysis
- performing probabilistic analysis

my research

A unified framework for

- writing specifications
- developing programs
- performing complexity analysis
- performing probabilistic analysis
- performing security analysis

my research

A unified framework for

- writing specifications
- developing programs
- performing complexity analysis
- performing probabilistic analysis
- performing security analysis

for

- (classical and) quantum algorithms

my research

A unified framework for

- writing specifications
- developing programs
- performing complexity analysis
- performing probabilistic analysis
- performing security analysis

for

- (classical and) quantum algorithms
- (classical and) quantum distributed systems

my research

A unified framework for

- writing specifications
- developing programs
- performing complexity analysis
- performing probabilistic analysis
- performing security analysis

for

- (classical and) quantum algorithms
- (classical and) quantum distributed systems
- (classical and) quantum communication protocols

my research

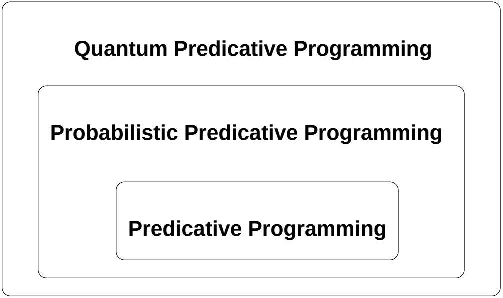
A unified framework for

- writing specifications
- developing programs
- performing complexity analysis
- performing probabilistic analysis
- performing security analysis

for

- (classical and) quantum algorithms
- (classical and) quantum distributed systems
- (classical and) quantum communication protocols
- (classical and) quantum cryptographic protocols

a theory of quantum programming



- predicative programming [H93,09]
- probabilistic predicative programming [H04,09]
- quantum predicative programming [T04],[TH06],[TH07,09],[TH09]

predicative programming

$$S \quad = \quad x \geq 0 \Rightarrow x' = 0$$

predicative programming

$$S \quad = \quad x \geq 0 \Rightarrow x' = 0$$

initial values : $x, y, \ldots$ final values : $x', y', \ldots$
specification : boolean expression
S is refined by P : $S \Leftarrow P$

predicative programming

$$S \quad = \quad x \geq 0 \Rightarrow x' = 0$$

$$S \quad \Longleftarrow \quad \textbf{if } x = 0 \textbf{ then } ok \textbf{ else } x := x - 1 \,;\, S$$

initial values : $x, y, \ldots$ final values : $x', y', \ldots$
specification : boolean expression
S is refined by P : $S \Leftarrow P$

predicative programming

$$S \quad = \quad x \geq 0 \Rightarrow x' = 0$$

$$S \quad \Longleftarrow \quad \textbf{if } x = 0 \textbf{ then } ok \textbf{ else } x := x - 1 \, ; \; S$$

$$\textbf{if } x = 0 \textbf{ then } ok \textbf{ else } x := x - 1 \, ; \; S$$

predicative programming

$$S \quad = \quad x \geq 0 \Rightarrow x' = 0$$

$$S \quad \Leftarrow \quad \textbf{if } x = 0 \textbf{ then } ok \textbf{ else } x := x - 1 \,;\, S$$

$$\begin{aligned} & \textbf{if } x = 0 \textbf{ then } ok \textbf{ else } x := x - 1 \,;\, S && S \text{ and } ok \\ = \; & \textbf{if } x = 0 \textbf{ then } x' = x \textbf{ else } x := x - 1 \,;\, x \geq 0 \Rightarrow x' = 0 \end{aligned}$$

$$ok = \; x' = x \wedge y' = y \wedge \ldots$$
$$x := e = \; x' = e \wedge y' = y \wedge \ldots$$

predicative programming

$$S \quad = \quad x \geq 0 \Rightarrow x' = 0$$

$$S \quad \Longleftarrow \quad \textbf{if } x = 0 \textbf{ then } ok \textbf{ else } x := x - 1 \,;\, S$$

$$
\begin{aligned}
& \textbf{if } x = 0 \textbf{ then } ok \textbf{ else } x := x - 1 \,;\, S && S \text{ and } ok \\
= \ & \textbf{if } x = 0 \textbf{ then } x' = x \textbf{ else } x := x - 1 \,;\, x \geq 0 \Rightarrow x' = 0 && \text{substitute} \\
= \ & \textbf{if } x = 0 \textbf{ then } x' = x \textbf{ else } x - 1 \geq 0 \Rightarrow x' = 0
\end{aligned}
$$

$$
\begin{aligned}
S \,;\, R = \ & \exists x'', y'', \ldots \cdot \text{for } x', y', \ldots \text{ substitute } x'', y'', \ldots \text{ in } S \\
& \qquad \wedge \text{ for } x, y, \ldots \text{ substitute } x'', y'', \ldots \text{ in } R \\
x := e \,;\, P = \ & \text{for } x \text{ substitute } e \text{ in } P
\end{aligned}
$$

predicative programming

$$S \quad == \quad x \geq 0 \Rightarrow x' = 0$$

$$S \quad \Longleftarrow \quad \textbf{if } x = 0 \textbf{ then } ok \textbf{ else } x := x - 1\ ;\ S$$

$\quad$ **if** $x = 0$ **then** ok **else** $x := x - 1\ ;\ S$ $\qquad$ S and ok

$==$ **if** $x = 0$ **then** $x' = x$ **else** $x := x - 1\ ;\ x \geq 0 \Rightarrow x' = 0$ substitute

$==$ **if** $x = 0$ **then** $x' = x$ **else** $x - 1 \geq 0 \Rightarrow x' = 0$ $\qquad$ if

$== x = 0 \wedge x' = x \vee x \neq 0 \wedge (x - 1 \geq 0 \Rightarrow x' = 0)$

$$\textbf{if } b \textbf{ then } S \textbf{ else} R == b \wedge S \vee \neg b \wedge R$$

predicative programming

$$S \quad = \quad x \geq 0 \Rightarrow x' = 0$$

$$S \quad \Longleftarrow \quad \textbf{if } x = 0 \textbf{ then } ok \textbf{ else } x := x - 1 \; ; \; S$$

$$
\begin{array}{lll}
& \textbf{if } x = 0 \textbf{ then } ok \textbf{ else } x := x - 1 \; ; \; S & S \text{ and } ok \\
= & \textbf{if } x = 0 \textbf{ then } x' = x \textbf{ else } x := x - 1 \; ; \; x \geq 0 \Rightarrow x' = 0 & \text{substitute} \\
= & \textbf{if } x = 0 \textbf{ then } x' = x \textbf{ else } x - 1 \geq 0 \Rightarrow x' = 0 & \text{if} \\
= & x = 0 \wedge x' = x \vee x \neq 0 \wedge (x - 1 \geq 0 \Rightarrow x' = 0) & \text{logic} \\
\Longrightarrow & x \geq 0 \Rightarrow x' = 0 & \\
= & S &
\end{array}
$$

quantum predicative programming

state:

$$\psi : 0,..2^n \to \mathbb{C}$$

$$\sum x : 0,..2^n \cdot |\psi x|^2 = 1$$

- Dirac-like notation:
  - $\mathbf{x}$: n-bit binary representation of x
  - $|\mathbf{x}\rangle = \lambda i : 0,..2^n \cdot (i = x)$
  - $|0\rangle = \lambda i : 0, 1 \cdot i = 0$
  - $|0\rangle/\sqrt{2} + |1\rangle/\sqrt{2}$
  - $|01\rangle = |0\rangle \otimes |1\rangle$
  - $|00\rangle/\sqrt{2} + |11\rangle/\sqrt{2}$

quantum predicative programming

operations on an n-qubit system ψ:

- initialisation to zero:

$$\psi := |0\rangle^{\otimes n}$$

- unitary transformation:

$$\psi := U\psi, \text{ where } U^\dagger U = I^n$$

quantum predicative programming

operations on an n-qubit system ψ:

- measurement:

$$\textbf{measure } \psi \ r \ \mathrel{=\!=} \ |\psi r'|^2 \times (\psi' = |\mathbf{r}'\rangle) \times (x' = x) \times (y' = y) \ldots$$

  - distribution of r' is $|\psi r'|^2$
  - distribution of the quantum state is

$$\sum r' \cdot |\psi r'|^2 \times (\psi' = |\mathbf{r}'\rangle)$$

  a *mixed* state

quantum predicative programming

a distributed $(n+m)$-qubit system ψ:

- unitary transformations:

$$
\begin{aligned}
\text{if } P &== \psi_{0,..n} := U_P \psi_{0,..n} \\
\text{and } Q &== \psi_{n,..n+m} := U_Q \psi_{n,..n+m} \\
\text{then } P \parallel_\psi Q &== \psi := (U_P \otimes U_Q)\psi
\end{aligned}
$$

- measurements:

$$
\begin{aligned}
\text{if } P &== \textbf{measure}_{M_P}\ \psi_{0,..n}\ p \\
\text{and } Q &== \textbf{measure}_{M_Q}\ \psi_{n,..n+m}\ q \\
\text{then } P \parallel_\psi Q &== \textbf{measure}_{M_P \otimes M_Q}\ \psi\ pq
\end{aligned}
$$

quantum predicative programming

A one-way quantum communication channel q from P to Q:

- M_q: infinite message script
- T_q: infinite time script
- r_q: read cursor
- w_q: write cursor
- n_q: number of quantum bits sent

$$
\begin{aligned}
q!\psi \quad &== M_q w_q = \psi \wedge T_q w_q = t \wedge \\
&\quad\; w'_q = w_q + 1 \wedge n'_q = n_q + 1 \wedge \mathbf{var}'_P := \mathbf{var}_P \backslash \psi \\
q?\psi \quad &== \psi' = M_q r_q \wedge r'_q = r_q + 1 \wedge \mathbf{var}'_Q := \mathbf{var}_Q, \psi
\end{aligned}
$$

Deutsch's algorithm

Specification:

$$S \;==\; x' = f0 \oplus f1 \wedge t' = t + 1$$

for $f : 0, 1 \to 0, 1$.

Program:

$$P == \psi := |0\rangle \; ; \; \psi := H\psi \; ; \; \psi := U_f \psi \; ;$$
$$\psi := H\psi \; ; \; \textbf{measure } \psi \; x$$

Prove: $S \Longleftarrow \; P$

Deutsch's algorithm

Specification:

$$S \Longleftarrow\!\!\!= \;\; x' = f0 \oplus f1 \wedge t' = t + 1$$

for $f : 0, 1 \to 0, 1$.

Program:

$$P == \psi := |0\rangle \; ; \; \psi := H\psi \; ; \; \psi := U_f \psi \; ; \; \psi := H\psi \; ; \; \textbf{measure} \; \psi \; x$$

Prove: $S \Longleftarrow P \longleftarrow$ machine verifiable

more algorithms

- Deutsch-Jozsa algorithm
- Grover's search
- . . .

see [T04], [TH06]

pseudo-telepathy games

A pseudo-telepathy game with n players:

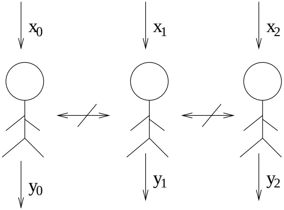
- D_i : domain of player i
- R_i : range of player i
- P : promise ($P = 1$, if no promise)
- W : winning condition
- S : strategy (program)

S is winning if

$$S\,!\,P \leq W\,!\,1$$

where $A\,!\,b \Longleftarrow (A \times b')/(A\;;\;b)$

Mermin's game



$$D_i == R_i == 0,1$$

$$P == x_0 \oplus x_1 \oplus x_2 = 0$$

$$W == (y_0' \oplus y_1' \oplus y_2') = (x_0 + x_1 + x_2)/2$$

Mermin's game

Program:

$$S == \psi := |000\rangle/\sqrt{2} + |111\rangle/\sqrt{2}\ ;\ S_0 \ ||_\psi\ S_1 \ ||_\psi\ S_2$$

$$S_i == \ \textbf{if}\ x_i = 1\ \textbf{then}\ \psi_i := U\psi_i\ \textbf{else}\ ok\ ;$$
$$\psi_i := H\psi_i\ ;\ \textbf{measure}\ \psi_i\ y_i$$

where $U|0\rangle = |0\rangle$ and $U|1\rangle = \sqrt{-1} \times |1\rangle$

Prove: $S\,!\,P \le W\,!\,1$

Mermin's game

Program:

$$S \Longleftarrow \psi := |000\rangle/\sqrt{2} + |111\rangle/\sqrt{2} \ ; \ S_0 \ ||_\psi \ S_1 \ ||_\psi \ S_2$$

$$S_i \Longleftarrow \textbf{if } x_i = 1 \textbf{ then } \psi_i := U\psi_i \textbf{ else } ok \ ;$$

$$\psi_i := H\psi_i \ ; \ \textbf{measure } \psi_i \ y_i$$

where $U|0\rangle = |0\rangle$ and $U|1\rangle = \sqrt{-1} \times |1\rangle$

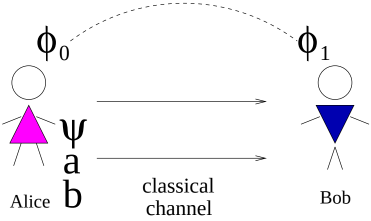
Prove: $S \,!\, P \leq W \,!\, 1 \longleftarrow$ machine verifiable

more pseudo-telepathy games

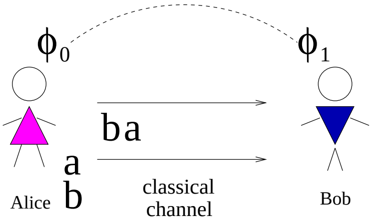
- Deutsch-Jozsa game
- parity games
- . . .

see [TH07]

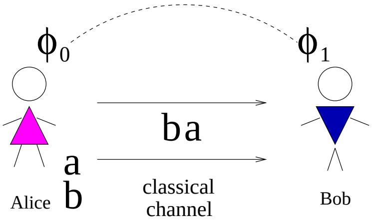
quantum teleportation



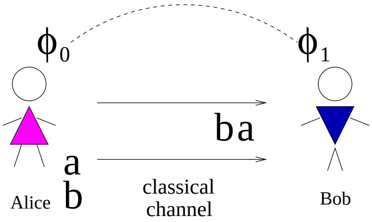
quantum teleportation



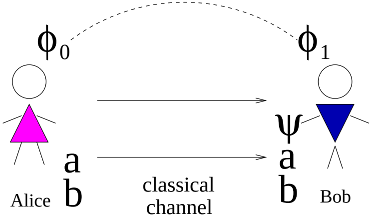
quantum teleportation



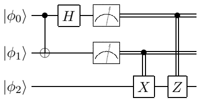
quantum teleportation



quantum teleportation



quantum teleportation



- Alice and Bob share $\phi_{12} = (|00\rangle + |11\rangle)/\sqrt{2}$
- Alice wants to teleport ϕ_0
- Alice performs local operations on ϕ_{01}, measures them, and sends results to Bob
- Bob performs local operations on ϕ_2, depending on the two classical bits received
- as a result, $\phi_2' = \phi_0$

quantum teleportation

Specification:

$$
\begin{aligned}
S \Longleftarrow\quad & \phi_{01} : \mathbf{var}_{Alice} \wedge \phi_2 : \mathbf{var}_{Bob} \wedge \\
& \phi_{0,1,2} = (\alpha \times |0\rangle + \beta \times |1\rangle) \otimes (|00\rangle + |11\rangle)/\sqrt{2} \\
\Rightarrow\; & (\phi_2' = \alpha \times |0\rangle + \beta \times |1\rangle) \wedge n_c' = n_c + 2 \wedge n_q' = n_q
\end{aligned}
$$

where

n_c number of classical bits sent

n_q number of quantum bits sent

quantum teleportation

Program:

$$P == \textbf{chan } ch : bit \cdot Alice_{a_0,a_1,\phi_{01}} \;||_{\phi}\; Bob_{b_0,b_1,\phi_2}$$

$$\text{where } Alice == \phi_{01} := CNOT\,\phi_{01} \;;\; \phi_0 := H\phi_0 \;;\; \textbf{measure } \phi_{01}\; a_0 a_1 \;;$$
$$ch\,!\,a_0 \;;\; ch\,!\,a_1$$

$$\text{and } Bob == ch? \;;\; b_0 := ch \;;\; ch? \;;\; b_1 := ch \;;\; \phi_2 := Z^{b_0} X^{b_1} \phi_2$$

Prove: $P \quad \leq \quad S$

quantum teleportation

Program:

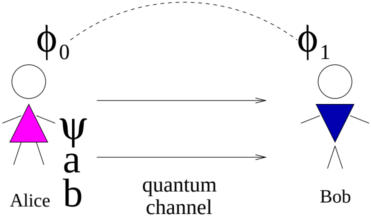
$$P \;=\!=\; \textbf{chan}\ ch : bit \cdot Alice_{a_0,a_1,\phi_{01}} \;||_\phi\; Bob_{b_0,b_1,\phi_2}$$

where $Alice \;=\!=\; \phi_{01} := CNOT\,\phi_{01}\ ;\ \phi_0 := H\phi_0\ ;\ \textbf{measure}\ \phi_{01}\ a_0 a_1\ ;$
$ch\,!\,a_0\ ;\ ch\,!\,a_1$

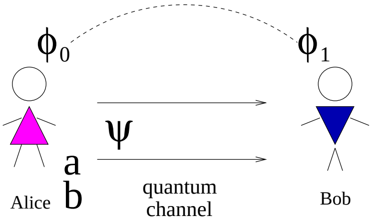
and $Bob \;=\!=\; ch?\ ;\ b_0 := ch\ ;\ ch?\ ;\ b_1 := ch\ ;\ \phi_2 := Z^{b_0} X^{b_1} \phi_2$

Prove: $P \quad \le \quad S \longleftarrow$ machine verifiable

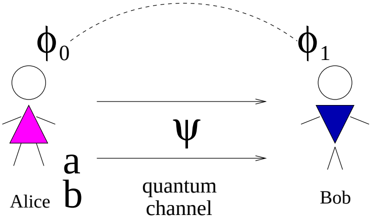
quantum super-dense coding



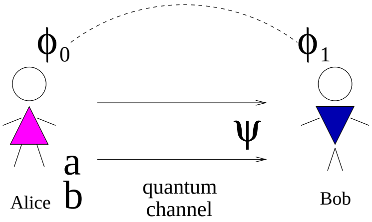
quantum super-dense coding



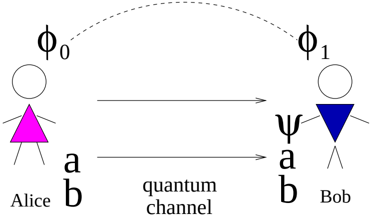
quantum super-dense coding



quantum super-dense coding



quantum super-dense coding



quantum super-dense coding

- Alice and Bob share $\phi = (|00\rangle + |11\rangle)/\sqrt{2}$
- Alice wants to send a_0, a_1
- Alice performs local operations on ϕ_0, depending on the two classical bits, and sends it to Bob
- Bob performs local operations on ϕ_{01} and measures them to obtain a_0 and a_1

quantum super-dense coding

Specification:

$$
\begin{aligned}
S \;==\quad & a_0, a_1, \phi_0 : \mathbf{var}_{Alice} \wedge b_0, b_1, \phi_1 : \mathbf{var}_{Bob} \wedge \\
& \phi_{01} = (|00\rangle + |11\rangle)/\sqrt{2} \\
\Rightarrow\; & b_0' = a_0 \wedge b_1' = a_1 \wedge n_c' = n_c \wedge n_q' = n_q + 1
\end{aligned}
$$

where

n_c number of classical bits sent

n_q number of quantum bits sent

quantum super-dense coding

Program:

$$\begin{aligned}
P &== \textbf{qchan}\ qch : qbit \cdot Alice_{a_0,a_1,\phi_0} \ ||_\phi\ Bob_{b_0,b_1,\phi_1} \\
\text{where } Alice &== \textbf{if}\ a_0 = a_1 = 0\ \textbf{then}\ ok \\
&\quad\ \textbf{else if}\ a_0 = 0 \wedge a_1 = 1\ \textbf{then}\ \phi_0 := X\phi_0 \\
&\quad\ \textbf{else if}\ a_0 = 1 \wedge a_1 = 0\ \textbf{then}\ \phi_0 := Z\phi_0 \\
&\quad\ \textbf{else}\ \phi_0 := Y\phi_0\ ; \\
&\quad\ qch\,!\,\phi_0 \\
\text{and } Bob &== qch?\phi_0\ ;\ \phi := CNOT\phi\ ;\ \phi_0 := H\phi_0\ ;\ \textbf{measure}\ \phi\ b_0 b_1
\end{aligned}$$

Prove: $P \quad \leq \quad S$

quantum super-dense coding

Program:

$$P \;=\!=\; \textbf{qchan}\; qch : qbit \cdot Alice_{a_0,a_1,\phi_0} \;||_\phi\; Bob_{b_0,b_1,\phi_1}$$

$$\text{where } Alice \;=\!=\; \textbf{if}\; a_0 = a_1 = 0 \;\textbf{then}\; ok$$

$$\textbf{else if}\; a_0 = 0 \wedge a_1 = 1 \;\textbf{then}\; \phi_0 := X\phi_0$$

$$\textbf{else if}\; a_0 = 1 \wedge a_1 = 0 \;\textbf{then}\; \phi_0 := Z\phi_0$$

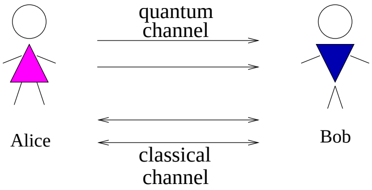
$$\textbf{else}\; \phi_0 := Y\phi_0\;;$$

$$qch\,!\,\phi_0$$

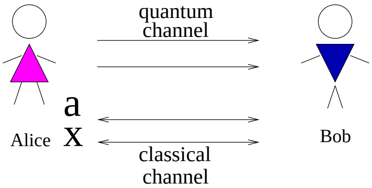
$$\text{and } Bob \;=\!=\; qch?\phi_0\;;\; \phi := CNOT\phi\;;\; \phi_0 := H\phi_0\;;\; \textbf{measure}\;\phi\; b_0 b_1$$

Prove: $P \;\leq\; S \longleftarrow$ machine verifiable

quantum key distribution – BB84

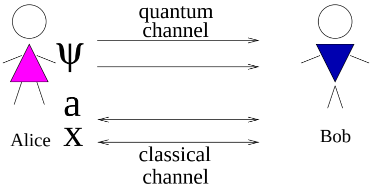


quantum key distribution – BB84



a,x – random

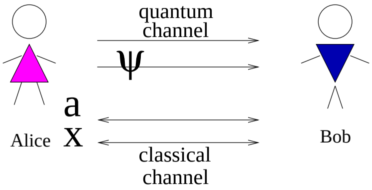
quantum key distribution – BB84



a,x – random

ψ – depend on a,x

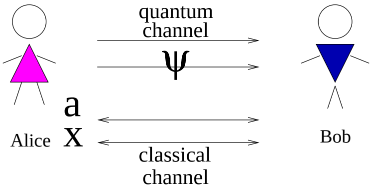
quantum key distribution – BB84



a,x – random

ψ – depend on a,x

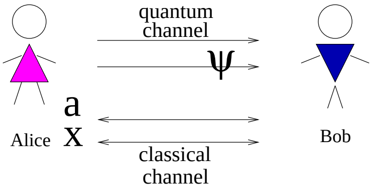
quantum key distribution – BB84



a,x – random

ψ – depend on a,x

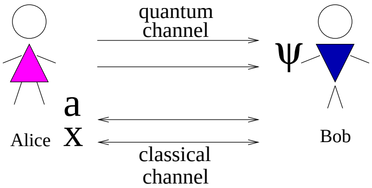
quantum key distribution – BB84



a,x – random

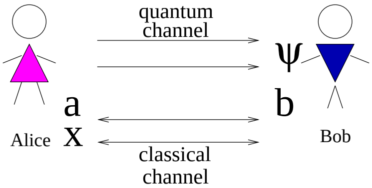
ψ – depend on a,x

quantum key distribution – BB84



a,x – random
ψ – depend on a,x

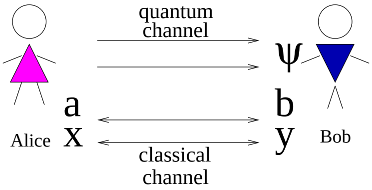
quantum key distribution – BB84



a,x − random   b − random

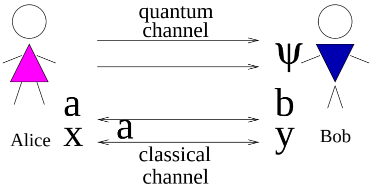
ψ − depend on a,x

quantum key distribution – BB84



a,x − random b − random
ψ − depend on a,x y − measurement result

quantum key distribution – BB84



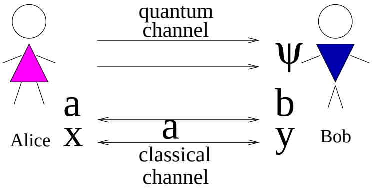
a,x – random

ψ – depend on a,x

b – random

y – measurement result

quantum key distribution – BB84



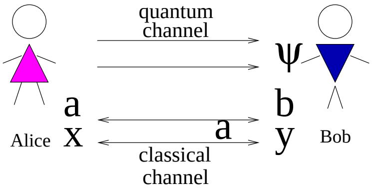
a,x – random

ψ – depend on a,x

b – random

y – measurement result

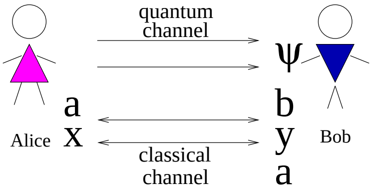
quantum key distribution – BB84



a,x − random   b − random

ψ − depend on a,x   y − measurement result

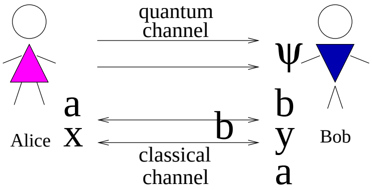
quantum key distribution – BB84



a,x − random b − random

ψ − depend on a,x y − measurement result

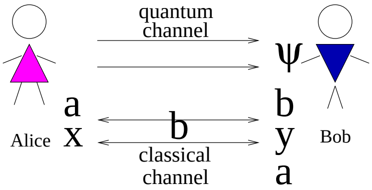
quantum key distribution – BB84



a,x – random $\qquad$ b – random

ψ – depend on a,x $\qquad$ y – measurement result

quantum key distribution – BB84



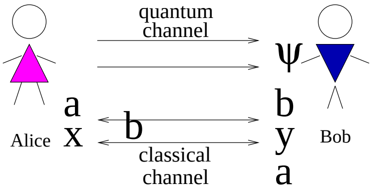
a,x – random

ψ – depend on a,x

b – random

y – measurement result

quantum key distribution – BB84



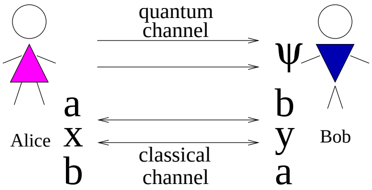
a,x – random

ψ – depend on a,x

b – random

y – measurement result

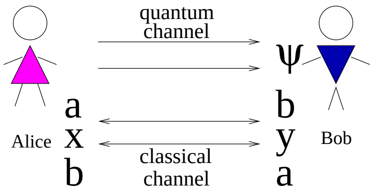
quantum key distribution – BB84



a,x − random
ψ − depend on a,x

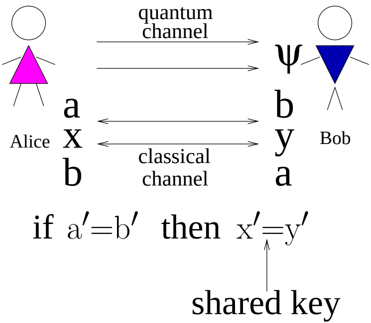
b − random
y − measurement result

quantum key distribution – BB84

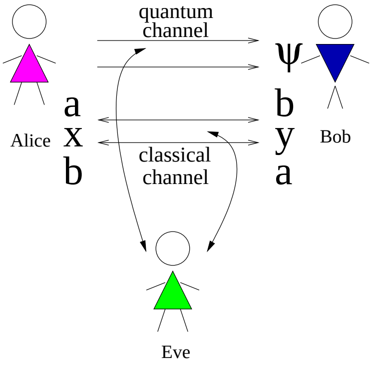


if a′=b′ then x′=y′

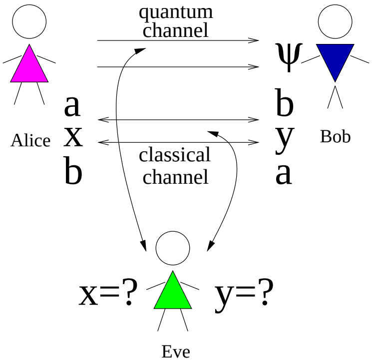
quantum key distribution – BB84



quantum key distribution – BB84



quantum key distribution – BB84



quantum key distribution – BB84

Program:

$$
\begin{aligned}
P &== \textbf{qchan}\ q : qbit \cdot \textbf{chan}\ c, d : bit \cdot Alice_{a,x,\psi,s_A} \,||\, Bob_{b,y,s_B} \\
\text{where } Alice &== (x' : 0,1)/2 \times (a' : 0,1)/2\ ;\ \psi := H^a|x\rangle\ ; \\
&\quad\ q!\psi\ ;\ c!a\ ;\ d?\ ;\ s_A := d \\
\text{and } Bob &== (b' : 0,1)/2\ ;\ q?\psi\ ;\ c?\ ;\ \textbf{measure}_b\ \psi\ y\ ; \\
&\quad\ s_B := (b = c)\ ;\ d!s_B
\end{aligned}
$$

Prove:

$$
\begin{aligned}
P\,!\,(s_A = 1) &\leq (x' = y')\,!\,1 \\
P\,!\,(s_B = 1) &\leq (x' = y')\,!\,1
\end{aligned}
$$

quantum key distribution – BB84

Program:

$$P == \textbf{qchan}\ q : qbit \cdot \textbf{chan}\ c, d : bit \cdot Alice_{a,x,\psi,s_A} \ || \ Bob_{b,y,s_B}$$

$$\text{where } Alice == (x' : 0,1)/2 \times (a' : 0,1)/2 \ ; \ \psi := H^a|x\rangle \ ;$$
$$q!\psi \ ; \ c!a \ ; \ d? \ ; \ s_A := d$$

$$\text{and } Bob == (b' : 0,1)/2 \ ; \ q?\psi \ ; \ c? \ ; \ \textbf{measure}_b\ \psi\ y \ ;$$
$$s_B := (b = c) \ ; \ d!s_B$$

Prove:

$$\left.\begin{array}{lll} P\,!\,(s_A = 1) & \leq & (x' = y')\,!\,1 \\ P\,!\,(s_B = 1) & \leq & (x' = y')\,!\,1 \end{array}\right\} \longleftarrow \text{machine verifiable}$$

more protocols

- n rounds of BB84
- BB92
- ...

see [T09]

summary

A unified framework for

- writing specifications
- developing programs
- performing complexity analysis
- performing probabilistic analysis
- performing security analysis

for

- (classical and) quantum algorithms
- (classical and) quantum distributed systems
- (classical and) quantum communication protocols
- (classical and) quantum cryptographic protocols

summary

A unified framework for

- writing specifications
- developing programs
- performing complexity analysis
- performing probabilistic analysis
- performing security analysis

for

- (classical and) quantum algorithms [T04,TH06]
- (classical and) quantum distributed systems [TH07,09]
- (classical and) quantum communication protocols [TH09]
- (classical and) quantum cryptographic protocols

summary

A unified framework for

- writing specifications
- developing programs
- performing complexity analysis
- performing probabilistic analysis
- performing security analysis

for

- (classical and) quantum algorithms [T04,TH06]
- (classical and) quantum distributed systems [TH07,09]
- (classical and) quantum communication protocols [TH09]
- (classical and) quantum cryptographic protocols[T09]

summary

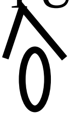
A unified framework for

- writing specifications
- developing programs
- performing complexity analysis
- performing probabilistic analysis
- performing security analysis

for

- (classical and) quantum algorithms [T04,TH06]
- (classical and) quantum distributed systems [TH07,09]
- (classical and) quantum communication protocols [TH09]
- (classical and) quantum cryptographic protocols[T09]

THANK YU !



references

T09 Tafliovich, A. *Distributed Quantum Computing*. In preparation 2009.

TH09 Tafliovich, A., Hehner, E.C.R.: *Programming with Quantum Communication*. In: QAPL 2009. Extended version to appear in ENTCS 2009.

TH07,09 Tafliovich, A., Hehner, E.C.R.: *Programming Telepathy: Implementing Quantum Non-locality Games*. In: SBFM 2007. Extended version to appear in ENTCS 2009.

TH06 Tafliovich, A., Hehner, E.C.R.: *Quantum Predicative Programming*. In: MPC 2006.

T04 Tafliovich, A.: *Quantum Programming*. Master's thesis, University of Toronto, 2004.

H93,09 Hehner, E.C.R.: *a Practical Theory of Programming*. Springer, New York, 1st ed., 1993. Curr. ed. (2009) `www.cs.utoronto.ca/~hehner/aPToP`

H09 Hehner, E.C.R.: *a Probability Perspective*. In: Formal Aspects of Computing, 2009, to appear.

H04 Hehner, E.C.R.: *Probabilistic Predicative Programming*. In: MPC 2004.