Qtpi: Simulating (Concurrent) Quantum Systems

Richard Bornat

(Emeritus) Department of Computer Science, Middlesex University, London, UK

BCS FACS, 19th October 2021
To begin

*Communicating Quantum Processes* (Gay and Nagarajan, POPL 2005) describes a programming language.
To begin

*Communicating Quantum Processes* (Gay and Nagarajan, POPL 2005) describes a programming language.

Communicating Quantum Processes (Gay and Nagarajan, POPL 2005) describes a programming language.


Qtpi is an implementation of modified CQP, with a symbolic quantum calculator.
Communicating Quantum Processes (Gay and Nagarajan, POPL 2005) describes a programming language.


Qtpi is an implementation of modified CQP, with a symbolic quantum calculator.

It’s available on github at mdxtoc/qtpi, with lots of examples and documentation. Can do BB84 QKD and other protocols, Grover’s algorithm, W state calculation, and more.
What’s interesting?

1. Quantum stuff
2. Language
3. Symbolic calculator
4. Probabilistic execution
5. Resource accounting for qubits
6. Qbit collections
7. Overloaded operators
8. Sparse matrix tricks
9. Iterative constructs
10. Demos
A protocol agent can:
Protocol steps

A protocol agent can:

- obtain a qubit;
Protocol steps

A protocol agent can:

- obtain a qubit;
- put a qubit (or qubits) through a quantum gate;
Protocol steps

A protocol agent can:

- obtain a qubit;
- put a qubit (or qubits) through a quantum gate;
- measure a qubit;
Protocol steps

A protocol agent can:

- obtain a qubit;
- put a qubit (or qubits) through a quantum gate;
- measure a qubit;
- send or receive a qubit;
A protocol agent can:

- obtain a qubit;
- put a qubit (or qubits) through a quantum gate;
- measure a qubit;
- send or receive a qubit;
- send or receive a classical value, such as a list of numbers or bits;
**Protocol steps**

A protocol agent can:

- obtain a qubit;
- put a qubit (or qubits) through a quantum gate;
- measure a qubit;
- send or receive a qubit;
- send or receive a classical value, such as a list of numbers or bits;
- do some classical calculation.
Process notation

Based on Milner’s pi calculus. Very stark.

\[
\text{procdef} ::= p(x, \ldots, z) = P
\]
**Process notation**

Based on Milner’s pi calculus. Very stark.

\[
\text{procdef} ::= p(x, \ldots, z) = P \\
\text{P} ::= IO \cdot P \mid qstep \cdot P \mid (\text{binder}) \cdot P
\]
Process notation

Based on Milner’s pi calculus. Very stark.

\[ \text{procdef} \ ::= \ p(x, \ldots, z) = P \]
\[ P \ ::= \ IO \cdot P \mid qstep \cdot P \mid (\text{binder}) \cdot P \]
\[ \quad \mid p(E, \ldots, E) \mid \text{par} \mid \text{alt} \mid \text{cond} \mid \_0 \]
**Process notation**

Based on Milner’s pi calculus. Very stark.

\[
\text{procdef} \ ::= \ p(x, \ldots, z) = P
\]

\[
P \ ::= \ IO \cdot P \mid qstep \cdot P \mid (\text{binder}) \cdot P
\quad \mid p(E, \ldots, E) \mid \text{par} \mid \text{alt} \mid \text{cond} \mid {}_0
\]

\[
IO \ ::= \ C ! E, \ldots, E \mid C ? (x, \ldots, z)
\]
Process notation

Based on Milner’s pi calculus. Very stark.

\[\text{procdef ::= } p(x, \ldots, z) = P\]

\[P ::= \text{IO} \cdot P \mid \text{qstep} \cdot P \mid (\text{binder}) \cdot P\]

\[\mid p(E, \ldots, E) \mid \text{par} \mid \text{alt} \mid \text{cond} \mid _0\]

\[\text{IO ::= } C ! E, \ldots, E \mid C ? (x, \ldots, z)\]

\[\text{qstep ::= } Q, \ldots, Q \gg G \mid q \vdash (x)\]
Process notation

Based on Milner’s pi calculus. Very stark.

\[
\text{procdef} ::= p(x, \ldots, z) = P \\
\]

\[
P ::= IO . P \mid qstep . P \mid (\text{binder}) . P \\
\]

\[
\mid p(E, \ldots, E) \mid \text{par} \mid \text{alt} \mid \text{cond} \mid \_0 \\
\]

\[
IO ::= C ! E, \ldots, E \mid C ? (x, \ldots, z) \\
\]

\[
qstep ::= Q, \ldots, Q \gg G \mid q \ll (x) \\
\]

\[
binder ::= \text{new } c \mid \text{newq } q \mid \text{newq } q = E \mid \text{let } \text{pat} = E \\
\]

Communication channels can be sent and received in messages; so can qubits. Qtpi is strongly typed (Hindley-Milner). Explicit typing is optional (but, for simplicity, not described).
Process notation

Based on Milner’s pi calculus. Very stark.

\[
\begin{align*}
\mathit{procdef} &::= p(x, \ldots, z) = P \\
\mathit{P} &::= \mathit{IO} . P \mid \mathit{qstep} . P \mid (\mathit{binder}) . P \\
&\quad \mid p(E, \ldots, E) \mid \mathit{par} \mid \mathit{alt} \mid \mathit{cond} \mid \_0 \\
\mathit{IO} &::= C!E, \ldots,E \mid C?\langle x, \ldots, z \rangle \\
\mathit{qstep} &::= Q, \ldots, Q \gg G \mid q\leftarrow(x) \\
\mathit{binder} &::= \mathit{new}c \mid \mathit{newq}q \mid \mathit{newq}q = E \mid \mathit{let} \mathit{pat} = E \\
\mathit{par} &::= [\mid \ldots \mid P \\
\mathit{alt} &::= [\mid + \mid \mathit{IO} . P + \ldots + \mathit{IO} . P \\
\mathit{cond} &::= \mathit{if}E \mathit{then} P \mathit{else} P \mid \mathit{match} E . \mathit{pat}.P + \ldots + \mathit{pat}.P
\end{align*}
\]

Communication channels can be sent and received in messages; so can qubits. Qtpi is strongly typed (Hindley-Milner). Explicit typing is optional (but, for simplicity, not described).
Process notation

Based on Milner’s pi calculus. Very stark.

\[
\begin{align*}
\text{procdef} & \ ::= \ p(x, \ldots, z) = P \\
\text{P} & \ ::= \ IO \ . \ P \mid qstep \ . \ P \mid (\text{binder}) \ . \ P \\
& \quad \mid p(E, \ldots, E) \mid \text{par} \mid \text{alt} \mid \text{cond} \mid _0 \\
\text{IO} & \ ::= \ C ! E, \ldots, E \mid C ? (x, \ldots, z) \\
\text{qstep} & \ ::= \ Q, \ldots, Q \gg G \mid q \not\triangleright (x) \\
\text{binder} & \ ::= \ \text{new} \ c \mid \text{newq} \ q \mid \text{newq} \ q = E \mid \text{let pat} = E \\
\text{par} & \ ::= \ [ \mid \text{P} \mid \ldots \mid \text{P} \\
\text{alt} & \ ::= \ [ + ] \ IO \ . \ P + \ldots + IO \ . \ P \\
\text{cond} & \ ::= \ \text{if } E \ \text{then } P \ \text{else } P \mid \text{match } E \ . \ \text{pat}.P + \ldots + \text{pat}.P
\end{align*}
\]

Communication channels can be sent and received in messages
Process notation

Based on Milner’s pi calculus. Very stark.

\[
\begin{align*}
\text{procdef} &::= p(x, \ldots, z) = P \\
\text{P} &::= IO \cdot P \mid qstep \cdot P \mid (\text{binder}) \cdot P \\
& \quad \mid p(E, \ldots, E) \mid \text{par} \mid \text{alt} \mid \text{cond} \mid _0 \\
\text{IO} &::= C! E, \ldots, E \mid C? (x, \ldots, z) \\
\text{qstep} &::= Q, \ldots, Q \triangleright G \mid q \not\triangleright (x) \\
\text{binder} &::= \text{new} c \mid \text{newq} \ q \mid \text{newq} \ q = E \mid \text{let} \ pat = E \\
\text{par} &::= [ \mid \mid \mid \mid P \mid \ldots \mid P \\
\text{alt} &::= [ + ] IO \cdot P + \ldots + IO \cdot P \\
\text{cond} &::= \text{if} E \text{then} P \text{else} P \mid \text{match} E \cdot \text{pat}.P + \ldots + \text{pat}.P
\end{align*}
\]

Communication channels can be sent and received in messages; so can qubits.
Process notation

Based on Milner’s pi calculus. Very stark.

\[
\begin{align*}
\text{procdef} &::= p(x, \ldots, z) = P \\
\quad P &::= IO \cdot P \mid qstep \cdot P \mid (\text{binder}) \cdot P \\
&\quad \mid p(E, \ldots, E) \mid \text{par} \mid \text{alt} \mid \text{cond} \mid _0 \\
\quad IO &::= C ! E, \ldots, E \mid C ? (x, \ldots, z) \\
\quad qstep &::= Q, \ldots, Q \gg G \mid q \rightleftharpoons (x) \\
\quad \text{binder} &::= \text{new} c \mid \text{newq} q \mid \text{newq} q = E \mid \text{let} \ pat = E \\
\quad \text{par} &::= [ \mid \mid P \mid \ldots \mid P \\
\quad \text{alt} &::= [ + ] IO \cdot P + \ldots + IO \cdot P \\
\quad \text{cond} &::= \text{if} E \text{ then } P \text{ else } P \mid \text{match} E \cdot pat.P + \ldots + pat.P
\end{align*}
\]

Communication **channels** can be sent and received in messages; so can qubits.

Qtpi is strongly typed (Hindley-Milner). Explicit typing is optional (but, for simplicity, not described).
There’s a functional notation for calculation, which looks like Miranda ...
Expression notation

There’s a functional notation for calculation, which looks like Miranda ... but it’s eager.
There’s a functional notation for calculation, which looks like Miranda ... but it’s eager.

To deal with qubit accounting, functions can’t have anything to do with qubits.
What’s interesting (again)?

1. Quantum stuff
2. Language
3. Symbolic calculator
4. Probabilistic execution
5. Resource accounting for qubits
6. Qbit collections
7. Overloaded operators
8. Sparse matrix tricks
9. Iterative constructs
10. Demos
Demo

Teleportation, queen of the baby protocols
BB84, queen of the QKD protocols

- generate a one-time code **without transmitting it.**
BB84, queen of the QKD protocols

- generate a one-time code without transmitting it.
- Alice chooses 1000 bits (say);
BB84, queen of the QKD protocols

generate a one-time code **without transmitting it.**
- Alice chooses 1000 bits (say);
- Alice sends them as 1000 qubits, randomly choosing diagonal (|+⟩, |−⟩) or normal (|0⟩, |1⟩) encoding for 0 and 1;
generate a one-time code **without transmitting it**.

- Alice chooses 1000 bits (say);
- Alice sends them as 1000 qubits, randomly choosing diagonal ($|+\rangle$, $|−\rangle$) or normal ($|0\rangle$, $|1\rangle$) encoding for 0 and 1;
- Bob measures them, randomly as diagonal or normal (50/50 he guesses right on each);
BB84, queen of the QKD protocols

- generate a one-time code without transmitting it.
- Alice chooses 1000 bits (say);
- Alice sends them as 1000 qubits, randomly choosing diagonal ($|+\rangle$, $|-\rangle$) or normal ($|0\rangle$, $|1\rangle$) encoding for 0 and 1;
- Bob measures them, randomly as diagonal or normal (50/50 he guesses right on each);
- They compare notes (classically) about their random choices;
BB84, queen of the QKD protocols

- generate a one-time code **without transmitting it**.
- Alice chooses 1000 bits (say);
- Alice sends them as 1000 qubits, randomly choosing diagonal ($|+\rangle$, $|-\rangle$) or normal ($|0\rangle$, $|1\rangle$) encoding for 0 and 1;
- Bob measures them, randomly as diagonal or normal (50/50 he guesses right on each);
- They compare notes (classically) about their random choices;
- If Eve has not intervened, they share $\sim 500$ secret bits;
**BB84, queen of the QKD protocols**

- generate a one-time code *without transmitting it*.
- Alice chooses 1000 bits (say);
- Alice sends them as 1000 qubits, randomly choosing diagonal (|+⟩, |−⟩) or normal (|0⟩, |1⟩) encoding for 0 and 1;
- Bob measures them, randomly as diagonal or normal (50/50 he guesses right on each);
- They compare notes (classically) about their random choices;
- If Eve has not intervened, they share ~500 secret bits;
- Bob sends Alice (classically) a random sample of \( n \) of his bits;
BB84, queen of the QKD protocols

- generate a one-time code **without transmitting it.**
- Alice chooses 1000 bits (say);
- Alice sends them as 1000 qubits, randomly choosing diagonal ($|+\rangle$, $|-\rangle$) or normal ($|0\rangle$, $|1\rangle$) encoding for 0 and 1;
- Bob measures them, randomly as diagonal or normal (50/50 he guesses right on each);
- They compare notes (classically) about their random choices;
- If Eve has not intervened, they share $\sim500$ **secret** bits;
- Bob sends Alice (classically) a random sample of $n$ of his bits;
- Only a $(\frac{3}{4})^n$ chance that Eve has meddled and those bits match;
BB84, queen of the QKD protocols

- generate a one-time code **without transmitting it**.
- Alice chooses 1000 bits (say);
- Alice sends them as 1000 qubits, randomly choosing diagonal ($|+\rangle$, $|-\rangle$) or normal ($|0\rangle$, $|1\rangle$) encoding for 0 and 1;
- Bob measures them, randomly as diagonal or normal (50/50 he guesses right on each);
- They compare notes (classically) about their random choices;
- If Eve has not intervened, they share $\sim$500 secret bits;
- Bob sends Alice (classically) a random sample of $n$ of his bits;
- Only a ($\frac{3}{4}$)$^n$ chance that Eve has meddled and those bits match;
- Otherwise A & B share a (500-$n$)-bit secret one-time code;
BB84, queen of the QKD protocols

- generate a one-time code **without transmitting it**.
- Alice chooses 1000 bits (say);
- Alice sends them as 1000 qubits, randomly choosing diagonal ($|+\rangle$, $|-\rangle$) or normal ($|0\rangle$, $|1\rangle$) encoding for 0 and 1;
- Bob measures them, randomly as diagonal or normal (50/50 he guesses right on each);
- They compare notes (classically) about their random choices;
- If Eve has not intervened, they share $\sim 500$ **secret** bits;
- Bob sends Alice (classically) a random sample of $n$ of his bits;
- Only a $(\frac{3}{4})^n$ chance that Eve has meddled and those bits match;
- Otherwise A & B share a $(500-n)$-bit **secret one-time code**;
- Alice uses it to XOR the message and send it classically.