Quantum CS with Graph Rewriting and CAS

Aleks Kissinger

Oxford University Computing Laboratory
alexander.kissinger@comlab.ox.ac.uk

July 6, 2009
Overview

- The standard opening line: “Quantum mechanics is hard.”
Overview

- The standard opening line: “Quantum mechanics is hard.”
- But...graphs make it easier
The standard opening line: “Quantum mechanics is hard.”
But...graphs make it easier
Graphical representation for quantum systems, and a graph rewrite system that gives a rich description of their theory
Overview

- The standard opening line: “Quantum mechanics is hard.”
- But...graphs make it easier
- Graphical representation for quantum systems, and a graph rewrite system that gives a rich description of their theory
- Expanding that theory usually takes lots of hard matrix work. Ideally we hand most of that off to a CAS
Overview

- The standard opening line: “Quantum mechanics is hard.”
- But...graphs make it easier
- Graphical representation for quantum systems, and a graph rewrite system that gives a rich description of their theory
- Expanding that theory usually takes lots of hard matrix work. Ideally we hand most of that off to a CAS
- Quantomatic bridges the gap between graph rewrite theories and CAS work
Hilbert Space Quantum Mechanics

- Pure state quantum mechanics has:

  - States: Elements of a Hilbert space $v \in H$
  - State evolutions: Unitary maps $u U^\dagger = U U^\dagger = \eta$
  - Observables: Self-adjoint $o = o^\dagger$ linear maps
  - Measurement: Sets of projections summing to the identity
  - Composite states: tensor product $v_1 \otimes v_2$

- Mixed state quantum mechanics has generalisations of the above. We won’t talk about that.

Hilbert Space Quantum Mechanics

- Pure state quantum mechanics has:
  - **States**: Elements of a Hilbert space, $\nu \in \mathcal{H}$
    - In finite dimensions, just think of these as plain old vector spaces with a dot product.
Pure state quantum mechanics has:

- **States**: Elements of a Hilbert space, $v \in \mathcal{H}$
  - In finite dimensions, just think of these as plain old vector spaces with a dot product.

- **State evolutions**: Unitary maps ($U^\dagger U = UU^\dagger = 1$)

Mixed state quantum mechanics has generalisations of the above. We won't talk about that.
Pure state quantum mechanics has:

- **States**: Elements of a Hilbert space, \( \nu \in \mathcal{H} \)
  - In finite dimensions, just think of these as plain old vector spaces with a dot product.
- **State evolutions**: Unitary maps \( (U^{\dagger}U = UU^{\dagger} = 1) \)
- **Observables**: Self-adjoint \( (O = O^{\dagger}) \) linear maps
Pure state quantum mechanics has:

- **States**: Elements of a Hilbert space, \( v \in \mathcal{H} \)
  - In finite dimensions, just think of these as plain old vector spaces with a dot product.
- **State evolutions**: Unitary maps \( (U^\dagger U = UU^\dagger = 1) \)
- **Observables**: Self-adjoint \( (O = O^\dagger) \) linear maps
- **Measurement**: Sets of projections, summing to the identity
Pure state quantum mechanics has:

- **States**: Elements of a Hilbert space, $\nu \in \mathcal{H}$
  - In finite dimensions, just think of these as plain old vector spaces with a dot product.

- **State evolutions**: Unitary maps \( U^\dagger U = UU^\dagger = 1 \)

- **Observables**: Self-adjoint \( \mathcal{O} = \mathcal{O}^\dagger \) linear maps

- **Measurement**: Sets of projections, summing to the identity

- **Composite states**: tensor product $\nu_1 \otimes \nu_2$
Hilbert Space Quantum Mechanics

- Pure state quantum mechanics has:
  - **States**: Elements of a Hilbert space, \( v \in \mathcal{H} \)
    - In finite dimensions, just think of these as plain old vector spaces with a dot product.
  - **State evolutions**: Unitary maps \( (U^\dagger U = UU^\dagger = 1) \)
  - **Observables**: Self-adjoint \( (O = O^\dagger) \) linear maps
  - **Measurement**: Sets of projections, summing to the identity
  - **Composite states**: tensor product \( v_1 \otimes v_2 \)

- Mixed state quantum mechanics has generalisations of the above. We won't talk about that.
Entanglement and the Tensor

For our purposes, take $\otimes$ to be the Kronecker product:

\[
\begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} ac \\ ad \\ bc \\ bd \end{pmatrix}
\]
Entanglement and the Tensor

For our purposes, take $\otimes$ to be the Kronecker product:

\[
\begin{pmatrix}
a \\
b \\
\end{pmatrix} \otimes \begin{pmatrix}
c \\
d \\
\end{pmatrix} = \begin{pmatrix}
ac \\
ad \\
bc \\
bd \\
\end{pmatrix}
\]

For Hilbert spaces $\mathcal{H}_1$ and $\mathcal{H}_2$, we can construct $\mathcal{H}_1 \otimes \mathcal{H}_2 = \text{span} \{ v \otimes u : v \in \mathcal{H}_1, u \in \mathcal{H}_2 \}$.

- $\dim (\mathcal{H}_1 \otimes \mathcal{H}_2) = \dim \mathcal{H}_1 \cdot \dim \mathcal{H}_2$
Entanglement and the Tensor

Some states \( w \in \mathcal{H}_1 \otimes \mathcal{H}_2 \) can be written as \( v \otimes u \) for \( v \in \mathcal{H}_1, u \in \mathcal{H}_2 \). These states are called \textit{separable}. 
Entanglement and the Tensor

- Some states \( w \in \mathcal{H}_1 \otimes \mathcal{H}_2 \) can be written as \( v \otimes u \) for \( v \in \mathcal{H}_1, u \in \mathcal{H}_2 \). These states are called \textit{separable}.

- ...but most can’t. These are called \textit{entangled}. They can be expressed as some sum \( \sum v_i \otimes u_i \) and are very important for doing lots of “quantum-like” stuff like teleportation.
Entanglement and the Tensor

- Some states $w \in \mathcal{H}_1 \otimes \mathcal{H}_2$ can be written as $v \otimes u$ for $v \in \mathcal{H}_1, u \in \mathcal{H}_2$. These states are called \textit{separable}.

- ...but most can’t. These are called \textit{entangled}. They can be expressed as some sum $\sum v_i \otimes u_i$ and are very important for doing lots of “quantum-like” stuff like teleportation.

- The Hilbert space $\mathcal{Q} := \mathbb{C}^2$ is called the space of \textit{qubits}.
Entanglement and the Tensor

Some states $w \in \mathcal{H}_1 \otimes \mathcal{H}_2$ can be written as $v \otimes u$ for $v \in \mathcal{H}_1$, $u \in \mathcal{H}_2$. These states are called *separable*.

...but most can’t. These are called *entangled*. They can be expressed as some sum $\sum v_i \otimes u_i$ and are very important for doing lots of “quantum-like” stuff like teleportation.

The Hilbert space $Q := \mathbb{C}^2$ is called the space of *qubits*.

We write the standard basis of $Q$ in “ket” notation, as $|0\rangle, |1\rangle$. Also, $|ij\rangle$ is shorthand for $|i\rangle \otimes |j\rangle$. 
We can think of unitary maps as the quantum analogy to reversible logic gates.
Quantum Circuits

- We can think of unitary maps as the quantum analogy to reversible logic gates.
- As such, we can dig right in into making some circuits.
Quantum Circuits

- We can think of unitary maps as the quantum analogy to reversible logic gates.
- As such, we can dig right in into making some circuits.
- First, some gates that are pretty “classical”
Quantum Circuits

- We can think of unitary maps as the quantum analogy to reversible logic gates.
- As such, we can dig right in into making some circuits.
- First, some gates that are pretty “classical”
  - NOT gates, \( \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \)
Quantum Circuits

- We can think of unitary maps as the quantum analogy to reversible logic gates.
- As such, we can dig right in into making some circuits.
- First, some gates that are pretty “classical”
  - NOT gates, \[
  \begin{pmatrix}
  0 & 1 \\
  1 & 0
  \end{pmatrix}
  \]
  - Controlled-NOT gates, \[
  \begin{pmatrix}
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & 0 & 1 \\
  0 & 0 & 1 & 0
  \end{pmatrix}
  \]
Quantum Circuits

► We can think of unitary maps as the quantum analogy to reversible logic gates.
► As such, we can dig right in into making some circuits.
► First, some gates that are pretty “classical”
  ► NOT gates, \[
  \begin{pmatrix}
  0 & 1 \\
  1 & 0
  \end{pmatrix}
  \]
  ► Controlled-NOT gates, \[
  \begin{pmatrix}
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & 0 & 1 \\
  0 & 0 & 1 & 0
  \end{pmatrix}
  \]
► And also some gates that are pretty “quantum”
Quantum Circuits

- We can think of unitary maps as the quantum analogy to reversible logic gates.
- As such, we can dig right in into making some circuits.
- First, some gates that are pretty “classical”
  - NOT gates, 
    \[
    \begin{pmatrix}
    0 & 1 \\
    1 & 0
    \end{pmatrix}
    \]
  - Controlled-NOT gates,
    \[
    \begin{pmatrix}
    1 & 0 & 0 & 0 \\
    0 & 1 & 0 & 0 \\
    0 & 0 & 0 & 1 \\
    0 & 0 & 1 & 0
    \end{pmatrix}
    \]
- And also some gates that are pretty “quantum”
  - Hadmard gates, 
    \[
    H = \frac{1}{\sqrt{2}} \begin{pmatrix}
    1 & 1 \\
    1 & -1
    \end{pmatrix}
    \]
Quantum Circuits

- We can think of unitary maps as the quantum analogy to reversible logic gates.
- As such, we can dig right in into making some circuits.
- First, some gates that are pretty “classical”
  - NOT gates, \( \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \)
  - Controlled-NOT gates, 
    \[
    \begin{pmatrix}
    1 & 0 & 0 & 0 \\
    0 & 1 & 0 & 0 \\
    0 & 0 & 0 & 1 \\
    0 & 0 & 1 & 0
    \end{pmatrix}
    \]
- And also some gates that are pretty “quantum”
  - Hadmard gates, \( H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \)
  - Phase gates, \( Z_\alpha = \begin{pmatrix} 1 & 0 \\ 0 & \text{e}^{i\alpha} \end{pmatrix} \)
Tensor product is represented by putting components side by side.

Matrix multiplication is graph composition.

Using these, we can build:

- A qubit swap: $\text{CNOT} \circ \text{CNOT}$
- An entangled state preparer: $\text{CNOT} \circ (H \otimes \text{y})$
- A NOT gate: $H \circ Z_{\pi} \circ H$

And lots of other stuff.
In Pictures

Tensor product is represented by putting components side-by-side. Matrix multiplication is graph composition. Using these, we can build...
Tensor product is represented by putting components side-by-side. Matrix multiplication is graph composition. Using these, we can build...

- A qubit swap \((CNOT \circ CNOT' \circ CNOT)\):
Tensor product is represented by putting components side-by-side. Matrix multiplication is graph composition. Using these, we can build:

- A qubit swap \((CNOT \circ CNOT' \circ CNOT)\):

- An entangled state preparer \((CNOT \circ (H \otimes 1))\):
Tensor product is represented by putting components side-by-side. Matrix multiplication is graph composition. Using these, we can build...

- A qubit swap \((CNOT \circ CNOT' \circ CNOT)\):

- An entangled state preparer \((CNOT \circ (H \otimes 1))\):

- A NOT gate \((H \circ Z_\pi \circ H)\):
Tensor product is represented by putting components side-by-side. Matrix multiplication is graph composition. Using these, we can build...

- A qubit swap \((\text{CNOT} \circ \text{CNOT}' \circ \text{CNOT})\):

- An entangled state preparer \((\text{CNOT} \circ (H \otimes 1))\):

- A NOT gate \((H \circ Z_\pi \circ H)\):

- ...and lots of other stuff
More Primitive

- So, what’s a CNOT, really?
More Primitive

So, what’s a CNOT, really?

copy the control qubit
More Primitive

- So, what’s a CNOT, really?

- copy the control qubit

- send one copy out
More Primitive

- So, what’s a CNOT, really?

  - copy the control qubit
  - “fuse” one copy here
  - send one copy out
Classical Structures

- A chosen basis is like some classical data embedded in the system.
Classical Structures

- A chosen basis is like some classical data embedded in the system.
- What can we do with classical data?
  - Copy and delete!

\[
\delta_Z : \mathbb{Q} \to \mathbb{Q} \otimes \mathbb{Q} :: |i\rangle \mapsto |ii\rangle \quad \epsilon_Z : \mathbb{Q} \to \mathbb{C} :: |i\rangle \mapsto 1
\]
A chosen basis is like some classical data embedded in the system.

What can we do with classical data?

- Copy and delete!

\[
\delta_Z : Q \rightarrow Q \otimes Q :: |i\rangle \mapsto |ii\rangle \quad \epsilon_Z : Q \rightarrow \mathbb{C} :: |i\rangle \mapsto 1
\]

Graphically:

\[
\delta_Z := \quad \epsilon_Z :=
\]
Classical Structures

- $\delta_Z$ has a (co)unit, $\epsilon_Z$:
Classical Structures

- $\delta_Z$ has a (co)unit, $\epsilon_Z$:

  $\delta_Z := \epsilon_Z$:

- $(-)^\dagger$ flips everything upside-down:

  $\delta_Z^\dagger := \epsilon_Z^\dagger$:
Classical Structures

- \( \delta_Z \) has a (co)unit, \( \epsilon_Z \):

- \((-)^{\dagger}\) flips everything upside-down:

- Phase gate \( Z_\alpha \) commutes with everything
Spiders

- Graphs of a single colour are extremely well behaved (associative, commutative, co-commutative, frobenius, etc...)

Graphs of a single colour are extremely well behaved (associative, commutative, co-commutative, frobenius, etc...)
In fact, they are uniquely determined by the number of inputs and outputs. As a result, we write connected graphs thus:
We can do the same thing for another basis:

\[ |+\rangle := \frac{1}{\sqrt{2}} \left( |0\rangle + |1\rangle \right) \quad |\rightarrow\rangle := \frac{1}{\sqrt{2}} \left( |0\rangle - |1\rangle \right) \]
We can do the same thing for another basis:

\[ |+\rangle := \frac{1}{\sqrt{2}} \left( |0\rangle + |1\rangle \right) \quad |\rightarrow\rangle := \frac{1}{\sqrt{2}} \left( |0\rangle - |1\rangle \right) \]

But, actually, there’s a shortcut. Realising the $H$ just interchanges the two bases:

\[ \cdots \alpha := \cdots \]

\[ \begin{array}{ccc}
\text{H} & \cdots & \text{H} \\
\cdots & \cdots & \cdots \\
\text{H} & \cdots & \text{H}
\end{array} \]
And we notice...

- We recover the bases, up to a scalar.

\[ \begin{align*}
0 & = |0\rangle + e^0 |1\rangle \approx |+\rangle \\
\pi & = |0\rangle + e^{i\pi} |1\rangle \approx |\rangle \\
0 & \approx |0\rangle \\
\pi & \approx |1\rangle
\end{align*} \]
And we notice...

- We recover the bases, up to a scalar.

\[
\begin{align*}
\bullet 0 &= |0\rangle + e^0 |1\rangle \approx |+\rangle \\
\bullet \pi &= |0\rangle + e^{i\pi} |1\rangle \approx |-\rangle \\
\bullet 0 &\approx |0\rangle \\
\bullet \pi &\approx |1\rangle
\end{align*}
\]

- These get copied and deleted, classical points.
And we notice...

- We recover the bases, up to a scalar.

\[
\begin{align*}
\text{Green } & \delta \text{'s copy red classical points and vice-versa.}
\end{align*}
\]

- These get copied and deleted, \textit{classical points}.

- Green $\delta'$s copy red classical points and vice-versa.
And we notice...

- We recover the bases, up to a scalar.
  
  \[
  0 = |0\rangle + e^0 |1\rangle \approx |+\rangle \quad \text{and} \quad \pi = |0\rangle + e^{i\pi} |1\rangle \approx |-\rangle
  \]

  \[
  0 \approx |0\rangle \quad \text{and} \quad \pi \approx |1\rangle
  \]

- These get copied and deleted, *classical points*.
- Green $\delta$'s copy red classical points and vice-versa.
- Red $(\delta_X, \epsilon_X)$ and green $(\delta_Z, \epsilon_Z)$ are *complementary classical structures*.
Rewrite Theory

sp: \[ \begin{array}{c}
\bullet \\
\bullet \\
\end{array} \begin{array}{c}
\circ \alpha \\
\circ \beta \\
\end{array} \rightarrow \begin{array}{c}
\bullet \\
\bullet \\
\end{array} \begin{array}{c}
\circ \alpha + \beta \\
\end{array} \]

el: \[ \begin{array}{c}
\circ \\
\rightarrow \\
\end{array} \]

tr: \[ \begin{array}{c}
\circ \alpha \\
\rightarrow \\
\circ \alpha \\
\end{array} \]
Rewrite Theory

sp: $\begin{array}{c}
\text{node} \\
\text{node}
\end{array}$ $\alpha$ $\begin{array}{c}
\text{node} \\
\text{node}
\end{array}$ $\beta$ $\rightarrow$ $\begin{array}{c}
\text{node}
\end{array}$ $\alpha + \beta$

el: $\begin{array}{c}
\text{node}
\end{array}$ $\rightarrow$

tr: $\begin{array}{c}
\text{node} \\
\text{node}
\end{array}$ $\alpha$ $\rightarrow$ $\begin{array}{c}
\text{node}
\end{array}$ $\alpha$

cc: $\begin{array}{c}
\text{node} \\
\text{node}
\end{array}$ $\alpha$ $\rightarrow$ $\begin{array}{c}
\text{node}
\end{array}$ $\alpha$
Rewrite Theory

sp: \[ \begin{array}{c}
\begin{array}{c}
\bullet \\
\bullet
\end{array}
\begin{array}{c}
\circ & \alpha \\
\circ & \beta
\end{array}
\end{array} \rightarrow
\begin{array}{c}
\begin{array}{c}
\bullet \\
\bullet
\end{array}
\begin{array}{c}
\circ & \alpha + \beta
\end{array}
\end{array} \]

el: \[ \begin{array}{c}
\begin{array}{c}
\circ
\end{array}
\rightarrow
\begin{array}{c}
\circ
\end{array}
\end{array} \]

tr: \[ \begin{array}{c}
\begin{array}{c}
\bullet
\end{array}
\begin{array}{c}
\circ & \alpha
\end{array}
\end{array} \rightarrow
\begin{array}{c}
\begin{array}{c}
\bullet
\end{array}
\begin{array}{c}
\circ & \alpha
\end{array}
\end{array} \]

cc: \[ \begin{array}{c}
\begin{array}{c}
\bullet
\end{array}
\begin{array}{c}
\circ & \alpha
\end{array}
\end{array} \rightarrow
\begin{array}{c}
\begin{array}{c}
\bullet
\end{array}
\begin{array}{c}
\circ & \alpha
\end{array}
\end{array} \]

ha: \[ \begin{array}{c}
\begin{array}{c}
\bullet
\end{array}
\begin{array}{c}
\circ & \alpha \\
\bullet & \beta
\end{array}
\end{array} \rightarrow
\begin{array}{c}
\begin{array}{c}
\bullet
\end{array}
\begin{array}{c}
\circ & \alpha
\end{array}
\end{array} \]

ba: \[ \begin{array}{c}
\begin{array}{c}
\bullet
\end{array}
\begin{array}{c}
\circ & \alpha \\
\circ & \beta
\end{array}
\end{array} \rightarrow
\begin{array}{c}
\begin{array}{c}
\bullet
\end{array}
\begin{array}{c}
\circ & \alpha
\end{array}
\end{array} \]
Doin’ the Swap
Quantomatic and Mathematica

- Rewrite theory is by design a course-graining
Quantomatic and Mathematica

- Rewrite theory is by design a course-graining
- Hybrid approach, graphical ↔ concrete
Quantomatic and Mathematica

- Rewrite theory is by design a course-graining
- Hybrid approach, graphical ↔ concrete
- For this, Quantomatic interfaces with Mathematica
Quantomatic and Mathematica

- Rewrite theory is by design a course-graining
- Hybrid approach, graphical ↔ concrete
- For this, Quantomatic interfaces with Mathematica
- Child process, utilises “everything is a term” design principle of Mathematica
Quanto comes up as child process. I then use the GUI to load a graph that gets named "Gb".
In[17]:= GetGraph["Gb"]

Out[17]= Gb
In[18]:= NormaliseGraph[

Out[18]=

Gf
In[20]:= GraphHilb[
  Graph[
    VertexList -> {0, 1},
    EdgeList -> {{0, 1}}
  ]
]

Out[20]= SparseArray[{<4>, {2, 2}}]

In[21]:= % // MatrixForm

Out[21]//MatrixForm=
\[
\begin{pmatrix}
1 & 1 \\
-1 & -1
\end{pmatrix}
\]
In[14]:= CheckRule

Out[14]= { }

Gd

Gc

0

(a)

- a
In[8]:= CheckRule

Out[8]= {{Quanto`Private`k -> 2}}
Rewriting ↔ CAS

- Normalising graphs first to make computations faster (or possible!)
Rewriting ↔ CAS

- Normalising graphs first to make computations faster (or possible!)
- Interplay of rewrite rules and semantics with CheckRule[], etc.
Rewriting ↔ CAS

- Normalising graphs first to make computations faster (or possible!)
- Interplay of rewrite rules and semantics with CheckRule[], etc.
- Numerics like entanglement measures, plots across free parameters
Entanglement Measures

\[ \pi / 3 \]

\[ q \text{tangle} \]

\[ 3\text{-tangle} \]
In[24]:= ThreeTangle

Out[24]= 0
In[33]:= PlotParam[ThreeTangle[], a]
Out[33]=

\[
\begin{align*}
\text{Out[33]} &= 0 \\
\frac{\pi}{3} &\quad \frac{2\pi}{3} &\quad \pi &\quad \frac{4\pi}{3} &\quad \frac{5\pi}{3} &\quad 2\pi \\
0.1 &\quad 0.2 &\quad 0.3 &\quad 0.4 &\quad 0.5 &\quad 0.6
\end{align*}
\]
Future Work

- Expand features, including a rule editor
- Rule feedback from CAS into Quantomatic
- Support other CAS’es, ideally use open-source alternatives
- Proper pattern graph matching, rather than “hacked” pattern graph matching
- Expand theory and solution techniques
Thanks!

- This is joint work with
  - Bob Coecke
    [http://www.comlab.ox.ac.uk/people/bob.coecke/](http://www.comlab.ox.ac.uk/people/bob.coecke/)
  - Ross Duncan
    [http://www.comlab.ox.ac.uk/people/ross.duncan/](http://www.comlab.ox.ac.uk/people/ross.duncan/)
  - Lucas Dixon

- Check it out at