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

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

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:

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.

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 ($U^\dagger U = UU^\dagger = 1$)

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

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

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 $v_1 \otimes v_2$

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 $\mathcal{H}_1 \otimes \mathcal{H}_2 = \operatorname{span}\{v \otimes u : v \in \mathcal{H}_1, u \in \mathcal{H}_2\}$.
  - $\dim(\mathcal{H}_1 \otimes \mathcal{H}_2) = \dim \mathcal{H}_1 \cdot \dim \mathcal{H}_2$

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 *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 *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.

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 $\mathcal{Q} := \mathbb{C}^2$ is called the space of *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 $\mathcal{Q} := \mathbb{C}^2$ is called the space of *qubits*.
- We write the standard basis of $\mathcal{Q}$ in "ket" notation, as $|0\rangle, |1\rangle$.

  Also, $|ij\rangle$ is shorthand for $|i\rangle \otimes |j\rangle$.

Quantum Circuits

- 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"

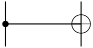
Quantum Circuits

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

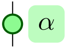
Quantum Circuits

- 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, $Z_\alpha = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\alpha} \end{pmatrix}$

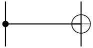
In Pictures

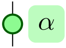
CNOT: 

H : 

Z_α : 

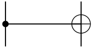
In Pictures

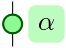
CNOT:  H : 

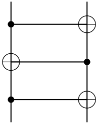
Z_α : 

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

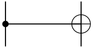
In Pictures

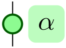
CNOT: 
H : 

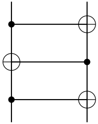
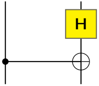
Z_α : 

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

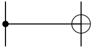
In Pictures

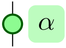
CNOT: 
H : 

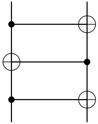
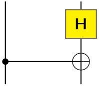
Z_α : 

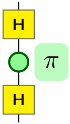
- Tensor product is represented by putting components side-by-side. Matrix multiplication is graph composition. Using these, we can build...
  - A qubit swap ($CNOT \circ CNOT' \circ CNOT$): 
  - An entangled state preparer ($CNOT \circ (H \otimes 1)$): 

In Pictures

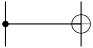
CNOT:  H : 

Z_α : 

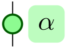
- Tensor product is represented by putting components side-by-side. Matrix multiplication is graph composition. Using these, we can build...
  - A qubit swap $(CNOT \circ CNOT' \circ CNOT)$: 
  - An entangled state preparer $(CNOT \circ (H \otimes 1))$: 

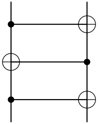
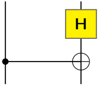
  - A NOT gate $(H \circ Z_\pi \circ H)$: 

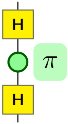
In Pictures

CNOT: 

H : 

Z_α : 

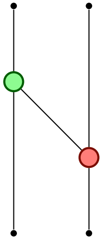
- Tensor product is represented by putting components side-by-side. Matrix multiplication is graph composition. Using these, we can build...
  - A qubit swap ($CNOT \circ CNOT' \circ CNOT$): 
  - An entangled state preparer ($CNOT \circ (H \otimes 1)$): 

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

  - ...and lots of other stuff

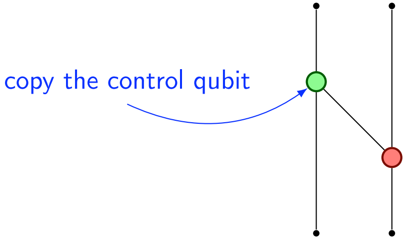
More Primitive

- So, what's a CNOT, really?



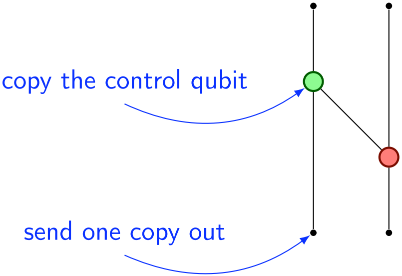
More Primitive

- So, what's a CNOT, really?



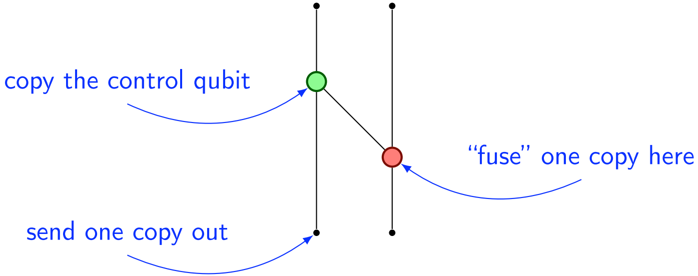
More Primitive

- So, what's a CNOT, really?



More Primitive

- So, what's a CNOT, really?



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!

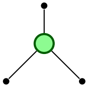
$$\delta_Z : \mathcal{Q} \to \mathcal{Q} \otimes \mathcal{Q} :: |i\rangle \mapsto |ii\rangle \qquad \epsilon_Z : \mathcal{Q} \to \mathbb{C} :: |i\rangle \mapsto 1$$

Classical Structures

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

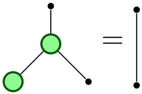
$$\delta_Z : \mathcal{Q} \to \mathcal{Q} \otimes \mathcal{Q} :: |i\rangle \mapsto |ii\rangle \qquad \epsilon_Z : \mathcal{Q} \to \mathbb{C} :: |i\rangle \mapsto 1$$

- Graphically:

$\delta_Z :=$  $\epsilon_Z :=$ 

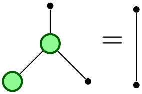
Classical Structures

- δ_Z has a (co)unit, ϵ_Z:

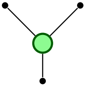


Classical Structures

- δ_Z has a (co)unit, ϵ_Z:

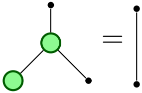


- $(-)^\dagger$ flips everything upside-down:

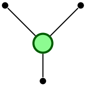
$\delta_Z^\dagger :=$  $\epsilon_Z^\dagger :=$ 

Classical Structures

- δ_Z has a (co)unit, ϵ_Z:



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

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

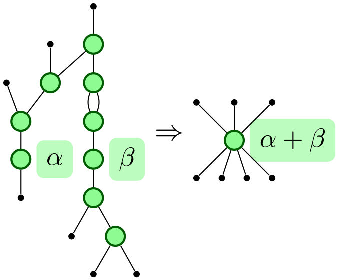
- Phase gate Z_α commutes with everything

Spiders

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

Spiders

- 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:



Another colour

- We can do the same thing for another basis:

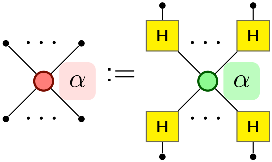
$$|+\rangle := \frac{1}{\sqrt{2}}\Big(|0\rangle + |1\rangle\Big) \qquad\qquad |-\rangle := \frac{1}{\sqrt{2}}\Big(|0\rangle - |1\rangle\Big)$$

Another colour

- We can do the same thing for another basis:

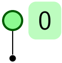
$$|+\rangle := \frac{1}{\sqrt{2}}\Big(|0\rangle + |1\rangle\Big) \qquad |-\rangle := \frac{1}{\sqrt{2}}\Big(|0\rangle - |1\rangle\Big)$$

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

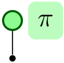


And we notice...

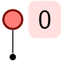
- We recover the bases, up to a scalar.



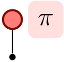
$= |0\rangle + e^{0}|1\rangle \approx |+\rangle$



$= |0\rangle + e^{i\pi}|1\rangle \approx |-\rangle$



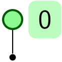
$\approx |0\rangle$



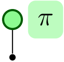
$\approx |1\rangle$

And we notice...

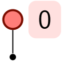
- We recover the bases, up to a scalar.



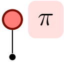
$= |0\rangle + e^{0}|1\rangle \approx |+\rangle$



$= |0\rangle + e^{i\pi}|1\rangle \approx |-\rangle$



$\approx |0\rangle$

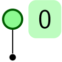


$\approx |1\rangle$

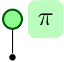
- These get copied and deleted, *classical points*.

And we notice...

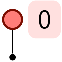
- We recover the bases, up to a scalar.



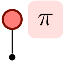
$= |0\rangle + e^{0}|1\rangle \approx |+\rangle$



$= |0\rangle + e^{i\pi}|1\rangle \approx |-\rangle$



$\approx |0\rangle$

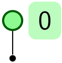


$\approx |1\rangle$

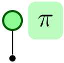
- These get copied and deleted, *classical points*.
- Green δ's copy red classical points and vice-versa.

And we notice...

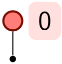
- We recover the bases, up to a scalar.



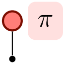
$= |0\rangle + e^{0}|1\rangle \approx |+\rangle$



$= |0\rangle + e^{i\pi}|1\rangle \approx |-\rangle$



$\approx |0\rangle$

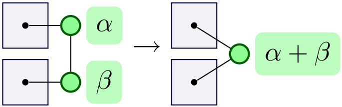


$\approx |1\rangle$

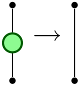
- These get copied and deleted, *classical points*.
- Green δ's copy red classical points and vice-versa.
- Red (δ_X, ϵ_X) and green (δ_Z, ϵ_Z) are *complementary classical structures*.

Rewrite Theory

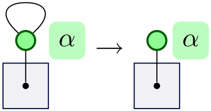
sp:



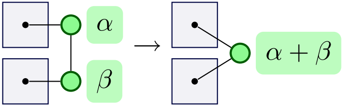
el:

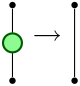


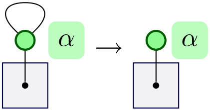
tr:

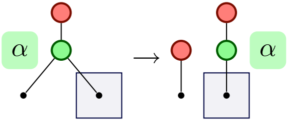


Rewrite Theory

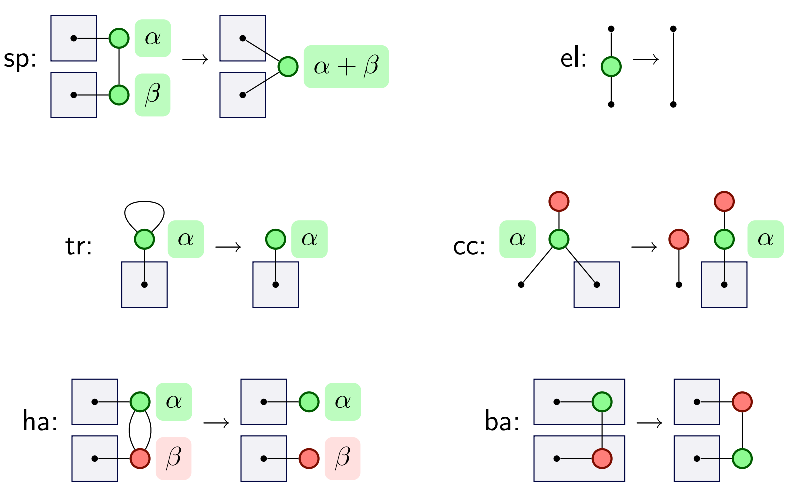
sp: 

el: 

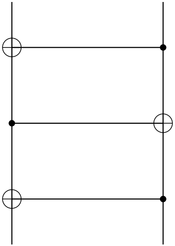
tr: 

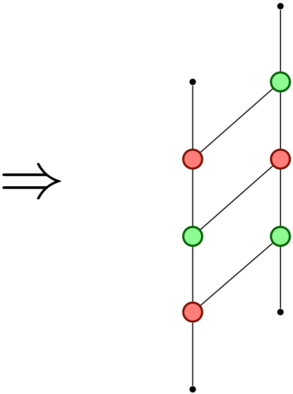
cc: 

Rewrite Theory



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 $\leftrightarrow$ concrete

Quantomatic and Mathematica

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

Quantomatic and Mathematica

- Rewrite theory is by design a course-graining
- Hybrid approach, graphical $\leftrightarrow$ concrete
- For this, Quantomatic interfaces with Mathematica
- Child process, utilises "everything is a term" design principle of Mathematica

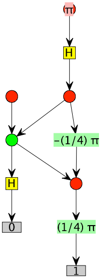
```
<< Quanto`

QuantoInit[
 "/Users/aleks/svn/isaplanner/quantomatic/gui/dist/
   QuantoGui.jar"]
```

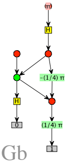
Quanto comes up as child process. I then use the GUI to load a graph that gets named "Gb".

In[17]:= `GetGraph["Gb"]`

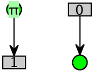
Out[17]=



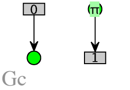
Gb

In[18]:= **NormaliseGraph**[]

Gb

Out[18]= 

Gf

In[20]:= `GraphHilb[`  `]`

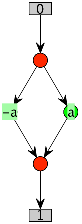
Out[20]= `SparseArray[<4>, {2, 2}]`

In[21]:= `% // MatrixForm`

Out[21]//MatrixForm=

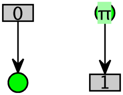
$$\begin{pmatrix} 1 & 1 \\ -1 & -1 \end{pmatrix}$$

In[14]:= **CheckRule**[



Gd

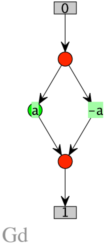
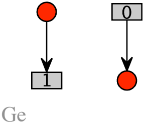
→



Gc

]

Out[14]= { }

In[8]:= **CheckRule**[ → ]

Out[8]= `{{Quanto`Private`k → 2}}`

Rewriting $\leftrightarrow$ CAS

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

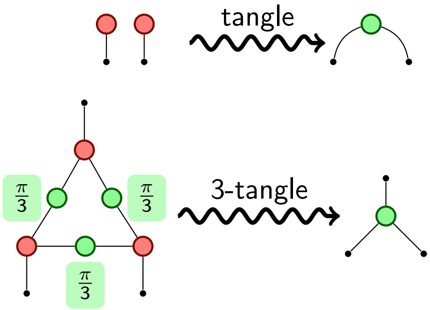
Rewriting $\leftrightarrow$ CAS

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

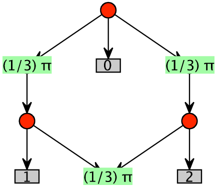
Rewriting $\leftrightarrow$ CAS

- 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



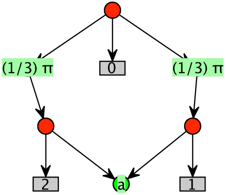
In[24]:= **ThreeTangle**[



]

Gg

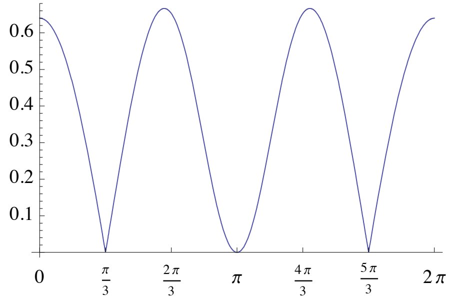
Out[24]= 0

In[33]:= **PlotParam[ThreeTangle[** 

], a]

Gg

Out[33]=



Future Work

- Expand features, including a rule editor
- Rule feed-back 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/
  - Ross Duncan
    http://www.comlab.ox.ac.uk/people/ross.duncan/
  - Lucas Dixon
    http://homepages.inf.ed.ac.uk/ldixon/
- Check it out at
  - http://dream.inf.ed.ac.uk/projects/quantomatic/