Tensors, !-graphs, and non-commutative quantum structures

Aleks Kissinger  David Quick

QPL
June 2014
<table>
<thead>
<tr>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
</tr>
<tr>
<td>Tensor notation</td>
</tr>
<tr>
<td>Definitions</td>
</tr>
<tr>
<td>Induction</td>
</tr>
<tr>
<td>Summary</td>
</tr>
</tbody>
</table>
Quanto
Defining Nodes

Given the graph:

Could we define:

\[ a \leftrightarrow c \leftrightarrow b \leftrightarrow a \]
Defining Nodes

Given the graph:

Could we define:
Defining Nodes

Given the graph:

Could we define:

\[
\begin{align*}
\text{Given the graph:} & \quad = \\
\text{Could we define:} & \quad :=
\end{align*}
\]
Defining Nodes

Given the graph:

Could we define:

Could we define:
Defining Nodes

Given the graph:

Could we define:

![Diagram](image)

Could we define:

![Diagram](image)
Defining Nodes

Given the graph:

Could we define:

\[
\begin{align*}
\text{a} & = \text{b} \\
\text{b} & = \text{a} \\
\text{c} & = \text{b}
\end{align*}
\]
Defining Nodes

Given the graph:

Could we define:

\[
\begin{align*}
\text{a} & \quad \text{c} \\
\text{b} & \quad \text{b} \\
\end{align*}
\]

:=

\[
\begin{align*}
\text{c} & \quad \text{b} \\
\text{a} & \quad \text{a} \\
\end{align*}
\]

Could we define:

\[
\begin{align*}
\text{a} & \quad \text{b} \\
\text{c} & \quad \text{c} \\
\end{align*}
\]

:=

\[
\begin{align*}
\text{a} & \quad \text{b} \\
\text{c} & \quad \text{c} \\
\end{align*}
\]

\[
\begin{align*}
\text{a} & \quad \text{b} \\
\text{c} & \quad \text{c} \\
\end{align*}
\]

\[
\begin{align*}
\text{a} & \quad \text{b} \\
\text{c} & \quad \text{c} \\
\end{align*}
\]

=
Defining Nodes

Given the graph:

Could we define:

\[
\begin{align*}
\text{c} & := \text{b} & \neq \text{a} \\
\text{b} & := \text{a} & \neq \text{c}
\end{align*}
\]
Defining Nodes

Given the graph:

Could we define:

\[
\begin{align*}
\begin{array}{c}
\text{a} \\
\text{b} \\
\text{c}
\end{array}
\end{align*}
\]

\[
\begin{align*}
\text{a} \\
\text{b} \\
\text{c}
\end{align*}
\]

\[
\begin{align*}
\text{a} \\
\text{b} \\
\text{c}
\end{align*}
\]

\[
\begin{align*}
\text{a} \\
\text{b} \\
\text{c}
\end{align*}
\]

\[
\begin{align*}
\text{a} \\
\text{b} \\
\text{c}
\end{align*}
\]

\[
\begin{align*}
\text{a} \\
\text{b} \\
\text{c}
\end{align*}
\]

\[
\begin{align*}
\text{a} \\
\text{b} \\
\text{c}
\end{align*}
\]
Recursively Defining Nodes

Given $\Sigma = \{\}$ satisfying:

\[
\begin{align*}
\begin{array}{c}
\vcenter{\hbox{\scalebox{0.5}{\input{example1.tex}}}} \\
\vcenter{\hbox{\scalebox{0.5}{\input{example2.tex}}}} \\
\vcenter{\hbox{\scalebox{0.5}{\input{example3.tex}}}} \\
\vcenter{\hbox{\scalebox{0.5}{\input{example4.tex}}}} \\
\end{array}
\end{align*}
\]

satisfying:

\[
\begin{align*}
\begin{array}{c}
\vcenter{\hbox{\scalebox{0.5}{\input{example1.tex}}}} \\
\vcenter{\hbox{\scalebox{0.5}{\input{example2.tex}}}} \\
\vcenter{\hbox{\scalebox{0.5}{\input{example3.tex}}}} \\
\vcenter{\hbox{\scalebox{0.5}{\input{example4.tex}}}} \\
\end{array}
\end{align*}
\]
Given $\Sigma = \{ \text{node forms} \}$ satisfying:

We may want to define nodes of the form:
Recursively Defining Nodes

0 inputs: $\uparrow := \downarrow$
Recursively Defining Nodes

0 inputs: $\vdots := \vdots$

Given $k$ inputs: $\vdots$
Recursively Defining Nodes

0 inputs:

Given k inputs:

define k+1 inputs:
Recursively Defining Nodes

0 inputs: $\bullet := \circ$

Given $k$ inputs:

define $k+1$ inputs: $\bullet := \circ$

Recursive definitions suggest proof by induction. To do this we need to be more formal with variable arity nodes.
Use !-boxes for multiple copies of a section from a graph.

![Diagram]

replaces
Use !-boxes for multiple copies of a section from a graph.

Then we have operations allowing deletion of !-boxes and creation of a new instance of the contents:
!-Boxes

Use !-boxes for multiple copies of a section from a graph.

Then we have operations allowing deletion of !-boxes and creation of a new instance of the contents:
Use !-boxes for multiple copies of a section from a graph.

Then we have operations allowing deletion of !-boxes and creation of a new instance of the contents:

- Kill
- Exp

![Diagram](https://example.com/diagram.png)
Use !-boxes for multiple copies of a section from a graph.

Then we have operations allowing deletion of !-boxes and creation of a new instance of the contents:
Use !-boxes for multiple copies of a section from a graph.

Then we have operations allowing deletion of !-boxes and creation of a new instance of the contents:
!-Box Equations

/-Boxes can be used in equations:
!-Box Equations

!-Boxes can be used in equations:

\[ \begin{array}{c}
\begin{array}{c}
\text{is replaced by}
\end{array}
\end{array} \]

\[ = \]

\[ = \]
!-Box Equations

!-Boxes can be used in equations:

\[
\begin{align*}
\cdots & = \cdots \quad \text{is replaced by} \quad \begin{array}{c}
\includegraphics[width=2cm]{example1.png}
\end{array} \\
& = \\
\end{align*}
\]

Which represents concrete equations like:

\[
\begin{align*}
\begin{array}{c}
\includegraphics[width=2cm]{example2.png}
\end{array} & = \begin{array}{c}
\includegraphics[width=2cm]{example3.png}
\end{array} \quad \begin{array}{c}
\includegraphics[width=2cm]{example4.png}
\end{array} & = \begin{array}{c}
\includegraphics[width=2cm]{example5.png}
\end{array} \quad \begin{array}{c}
\includegraphics[width=2cm]{example6.png}
\end{array} & = \begin{array}{c}
\includegraphics[width=2cm]{example7.png}
\end{array} \\
& \quad \cdots
\end{align*}
\]
This !-box (labelled A) has many different possible expansions:
This !-box (labelled A) has many different possible expansions:
This !-box (labelled A) has many different possible expansions:
This !-box (labelled A) has many different possible expansions:
This !-box (labelled A) has many different possible expansions:
This !-box (labelled A) has many different possible expansions:
This !-box (labelled A) has many different possible expansions:
This !-box (labelled A) has many different possible expansions:
This !-box (labelled A) has many different possible expansions:
Building a Tensor

\[
\psi_{a b} = \psi \hat{f}_{a \hat{b}} \hat{\psi}_{\hat{a} \hat{b}}
\]

\[
\phi_{a b c d e} = \phi_{\hat{a} \hat{b} \hat{c} \hat{d} \hat{e}}
\]
Building a Tensor

$$\psi = \psi_{\hat{f}}^{\hat{a}}_{\hat{b}} \hat{\psi}^{\hat{f}}_{\hat{a}} \hat{\psi}^{\hat{b}}_{\hat{a}}$$

$$\phi_{\hat{a}}_{\hat{b}}_{\hat{c}}_{\hat{d}}_{\hat{e}} = \phi^{\hat{a}}_{\hat{b}}^{\hat{c}}_{\hat{d}}_{\hat{e}}$$
Building a Tensor

\[ \psi_{fab} = \psi \]

\[ \phi_{ab} = \psi^{\hat{a}\hat{b}} \]
Building a Tensor

\[ \psi f_{\hat{a} \hat{b}} = \psi \]

\[ \phi_{a b c d e} = \phi_{\hat{a} \hat{b} \hat{c} \hat{d} \hat{e}} \]
Building a Tensor

\[ \psi_{\hat{a}\hat{b}} = \psi_{\hat{f}} \]

\[ \phi_{\hat{a}\hat{b}\hat{c}\hat{d}\hat{e}} = \phi_{\hat{a}} \]

Diagram:

- \( \psi_{\hat{a}\hat{b}} = \psi_{\hat{f}} \) with nodes \( a, b, f \)
- \( \phi_{\hat{a}\hat{b}\hat{c}\hat{d}\hat{e}} = \phi_{\hat{a}} \) with nodes \( a, b, c, d, e \)
Building a Tensor

\[ \psi_f \hat{a} \hat{b} \phi_\hat{a} \hat{b} \hat{c} \hat{d} \hat{e} = \psi_f \hat{a} \hat{b} \phi_\hat{a} \hat{b} \hat{c} \hat{d} \hat{e} \]
Building a Tensor

\[ \psi \hat{a} \hat{b} \phi \hat{a} \hat{b} \hat{c} \hat{d} \hat{e} = \psi \hat{f} \hat{a} \hat{b} \hat{c} \hat{d} \hat{e} \]
Building a Tensor

\[ \psi \hat{a} \hat{b} \hat{c} \hat{d} \hat{e} = \psi \hat{f} \hat{a} \hat{b} \hat{c} \hat{d} \hat{e} \]
Building a Tensor

\[ \psi_{fab} \phi_{abe} \]

\[ = \]

\[ \psi_{fabc} \phi_{abcde} \]

Diagram illustrating the tensors and their relationships.
Building a Tensor

\[ \psi_{\hat{a}\hat{b}\phi} \hat{a}\hat{b}c\hat{d}\hat{e} = \psi_{\hat{f}\hat{e}\phi} \hat{f}\hat{e}c\hat{d}\hat{e} = \psi_{\hat{x}\hat{y}\phi} \hat{x}\hat{y}c\hat{d}\hat{e} \]
Building a Tensor

\[ \psi_{\hat{a}\hat{b}} \phi_{\hat{a}\hat{b}\hat{c}\hat{d}\hat{e}} = \psi_{\hat{f}} \phi_{\hat{a}\hat{b}\hat{c}\hat{d}\hat{e}} \]

Diagram: A network with nodes labeled \( \phi \) and edges connecting them, representing the tensor product.
Building a Tensor

\[ [\psi \hat{f} \hat{a} \hat{b}]^B \phi \hat{a} \hat{b} \hat{c} \hat{d} \hat{e} = \]
Building a Tensor

\[
\begin{align*}
\left[ \psi_{\hat{f} \hat{a} \hat{b}} \right]^B \phi_{\hat{a} \hat{b} \hat{c} \hat{d} \hat{e}} &= \begin{array}{c}
\psi_{\hat{f} \hat{a} \hat{b}} \\
\phi_{\hat{a} \hat{b} \hat{c} \hat{d} \hat{e}}
\end{array}
\end{align*}
\]
Building a Tensor

\[
\left[ \psi_{\hat{a}\hat{b}} \right]^B \phi_{\langle \hat{a} \rangle}^B \hat{c} \hat{d} \hat{e} =
\]

\[
B
\]

\[
\phi
\]

\[
f \rightarrow \psi \leftarrow \phi \rightarrow c \rightarrow e \rightarrow d
\]
Building a Tensor

\[
\left[ \psi_{\hat{a}\hat{b}} \right]^B \phi_{\langle \hat{a}\hat{b}\rangle^B} =
\]

\[
\psi_{\hat{f}\hat{a}\hat{b}} \phi_{\langle \hat{a}\rangle^B} \hat{b} \hat{c} \hat{d} \hat{e}
\]
Building a Tensor

\[
\left[ \psi_{\hat{\alpha}\hat{\beta}} \right]^B \phi \langle \hat{\alpha} \rangle^B \hat{\beta}^{\hat{\alpha} \hat{\beta}} = \nabla
\]
Building a Tensor

\[
\left[ \psi_{\hat{a}\hat{b}} \right]^B \phi \langle \hat{a} \rangle^B [\hat{b}]^B \epsilon \hat{d} \hat{e} =
\]
An Example
An Example

\[ \psi [\hat{a}]^B \phi [\hat{a}]^B = \]

\begin{tikzpicture}
  \node (phi) at (0,0) {$\phi$};
  \node (psi) at (0,-1) {$\psi$};
  \draw[->] (phi) -- (psi);
  \node (b) at (0,-1.5) {B};
\end{tikzpicture}
An Example

\[ \psi_{[\hat{a}]^B} \phi_{[\hat{\alpha}]^B} [1]^B = \]

\[
\begin{array}{c}
\phi \\
\downarrow \\
B \\
\uparrow \\
\psi
\end{array}
\]
An Example

\[ \psi_{[\hat{a}]}^B \phi_{\langle \hat{a} \rangle}^B \begin{bmatrix} 1 \end{bmatrix}^B = \begin{bmatrix} \phi \end{bmatrix} \begin{bmatrix} \psi \end{bmatrix} \]

, , , , , ...

[Diagram with arrows between \( \phi \) and \( \psi \)]
Twisted Example

\[ \psi_{[\hat{\alpha}]} B \phi_{[\bar{\alpha}]} B [1] B = \]

\[ \phi \]

\[ \psi \]
Twisted Example

\[ \psi_{\hat{a}} B \phi_{\hat{a}} B [1]^B = \]

\[ \phi \]

\[ B \]

\[ \psi \]

\[ , \]

\[ , \]

\[ , \]

\[ , \]

\[ , \]

\[ , \]

\[ \phi \]

\[ , \]

\[ , \]

\[ , \]

\[ , \]

\[ , \]

\[ , \]

\[ , \]

\[ , \]

\[ , \]

\[ , \]
Nesting Example (Inner)

\[
\left[ \left[ \xi_{b \bar{a}} \right]^B \phi_{\langle b \rangle^B} \right]^A =
\]

\[\xi_{b \bar{a}} \phi_{\langle b \rangle^B}\]
Nesting Example (Inner)

\[
\left[ \left[ \xi_b^a \right]^B \phi^B \right]^A = \left[ \xi^a \right]^B \left[ \xi^b \right]^B \left[ \phi^B \right]^A
\]

\[\text{Kill}_B\]

\[\text{Exp}_B\]
Nesting Example (Inner)

\[
\left[ \left[ \xi_{\hat{b}a} \right]_B \phi_{\langle \tilde{b} \rangle B} \right]^A =
\]

\[
\begin{array}{c}
\phi \\
\end{array}
\]

\begin{array}{c}
\xi \\
a
\end{array}

\begin{array}{c}
\xi_{\hat{b}a} \\
\xi_{\hat{b}' a'} \phi_{\langle \tilde{b}' \rangle B}
\end{array}

\begin{array}{c}
a \\
a'
\end{array}

\begin{array}{c}
\phi \\
\end{array}

\begin{array}{c}
\xi \\
a
\end{array}

\begin{array}{c}
\xi_{\hat{b}a} \\
\xi_{\hat{b}' a'} \phi_{\langle \tilde{b}' \rangle B}
\end{array}

\begin{array}{c}
a \\
a'
\end{array}
Nesting Example (Outer)

\[
\left[\left[ \xi \hat{b} \right]^B \phi \xi^B \right]^A =
\]

\begin{center}
\begin{tikzpicture}
\node at (0,0) [circle,draw,fill=white,inner sep=0pt,minimum size=10pt] {\( \phi \)};
\node at (0,-1) [circle,draw,fill=white,inner sep=0pt,minimum size=10pt] {\( \xi \)};
\node at (0,-2) [circle,draw,fill=white,inner sep=0pt,minimum size=10pt] {a};
\end{tikzpicture}
\end{center}
Nesting Example (Outer)

\[
\left[ \left[ \xi \hat{b} \right]^B \phi \left( \hat{b} \right)^B \right]^A = \]

\[
\begin{array}{c}
\text{Kill}_A \\
\end{array}
\]

\[
\begin{array}{c}
\text{Exp}_A \\
\end{array}
\]
Nesting Example (Outer)

\[
\left[ \left[ \xi_{\bar{b}a} \right]_B^B \phi_{\langle \bar{b} \rangle}^B \right]^A = \\
\left[ \left[ \xi_{\bar{b}a} \right]_B^B \phi_{\langle \bar{b} \rangle}^B \right]_A
\]

\[
\text{Kill}_A \\
\text{Exp}_A
\]
Table of Contents

Introduction

Tensor notation

Definitions

Induction

Summary
Edgeterms

Definition
Fix disjoint, infinite sets $\mathcal{E}$ and $\mathcal{B}$ of edge names and $!$-box names, respectively. The set of edgeterms $\mathcal{T}_e$ is defined recursively as follows:

- $\epsilon \in \mathcal{T}_e$ (i.e. empty)
- $\check{a}, \hat{a} \in \mathcal{T}_e$  
- $\langle e \rangle^A, \langle e \rangle^A \in \mathcal{T}_e$  
- $[e]^A, [e]^A \in \mathcal{T}_e$  
- $ef \in \mathcal{T}_e$  
- $e, f \in \mathcal{T}_e$
Edgeterms

Definition
Fix disjoint, infinite sets $\mathcal{E}$ and $\mathcal{B}$ of edge names and $!$-box names, respectively. The set of edgeterms $\mathcal{T}_e$ is defined recursively as follows:

- $\epsilon \in \mathcal{T}_e$ (i.e empty)
- $\check{a}, \hat{a} \in \mathcal{T}_e$ $a \in \mathcal{E}$
- $[e]^A, \langle e \rangle^A \in \mathcal{T}_e$ $e \in \mathcal{T}_e, A \in \mathcal{B}$
- $ef \in \mathcal{T}_e$ $e, f \in \mathcal{T}_e$

Two edgeterms are equivalent if one can be transformed into the other by:

$$\epsilon e \equiv e \equiv e\epsilon \quad e(fg) \equiv (ef)g \quad [\epsilon]^A \equiv \epsilon \equiv \langle \epsilon \rangle^A$$
Definition
The set of all !-tensor expressions $\mathcal{T}_\Sigma$ for a signature $\Sigma$ is defined recursively as:

- $1 \in \mathcal{T}_\Sigma$ (empty tensor)
- $1_{\hat{a} \hat{b}} \in \mathcal{T}_\Sigma$ $a, b \in \mathcal{E}$
- $\phi_e \in \mathcal{T}_\Sigma$ $e \in \mathcal{T}_e, \phi \in \Sigma$
- $[G]^A \in \mathcal{T}_\Sigma$ $G \in \mathcal{T}_\Sigma, A \in \mathcal{B}$
- $GH \in \mathcal{T}_\Sigma$ $G, H \in \mathcal{T}_\Sigma$

Satisfying conditions F1-2, C1-3
Conditions: F1-F2

F1: No directed edge name can appear more than once as these can not be plugged together: \( \hat{\phi} \hat{\psi} = \)

F2: No box can appear more than once to prevent overlap (which we do not allow in this formalism):
Conditions: F1-F2

F1: No directed edge name can appear more than once as these can not be plugged together: $\phi \hat{a} \psi \hat{a}$
Conditions: F1-F2

F1: No directed edge name can appear more than once as these can not be plugged together: \( \phi \hat{a} \psi \hat{a} = \)

F2: No box can appear more than once to prevent overlap (which we do not allow in this formalism): 

\[
[[\ldots]^A] B \quad [\ldots]^A =
\]

\[
\begin{array}{c}
\text{A} \\
\text{A}
\end{array}
\]
Conditions: F1-F2

F1: No directed edge name can appear more than once as these can not be plugged together:  \[ \phi \hat{a} \psi \hat{a} = \]

F2: No box can appear more than once to prevent overlap (which we do not allow in this formalism):  \[ [[[\ldots]]^A]^B[[\ldots]]^A = \]
Conditions: C1-C3

C1: An edge entering a !-box can’t be on a node already in that !-Box:

\[
[\phi_{[\hat{a}\!]}^B] = \begin{array}{c}
\phi \\
\downarrow \\
B \\
\end{array}
\]

C2: Nested !-Boxes around an edge must be nested in the same way in the rest of the tensor:

\[
\phi_{[\hat{a}]}^A \phi_{[\hat{a}]}^B = \psi_A \psi_B
\]

C3: Bound edges must be in compatible !-boxes:

\[
\phi_{[\hat{a}]}^B \tilde{\phi}_{\hat{a}}^B = \psi_A \psi_B
\]
Conditions: C1-C3

C1: An edge entering a |-box can’t be on a node already in that |-Box:

\[ \left[ \phi_{[\hat{a}]}^B \right] = \]

\[ \phi \]

\[ \hat{a} \]

\[ B \]

\[ \text{a} \]

\[ B \]
Conditions: C1-C3

C1: An edge entering a |-box can’t be on a node already in that |-Box:

\[
\phi_{[\hat{\alpha}]_B}^B = \phi_{[\hat{\alpha}]_B^B}
\]

C2: Nested |-Boxes around an edge must be nested in the same way in the rest of the tensor:

\[
\phi_{[[\hat{\alpha}]_A^A}^B [\psi]^A = \phi_{[\hat{\alpha}]_A^A}^B [\psi]^A
\]
Conditions: C1-C3

C1: An edge entering a !-box can’t be on a node already in that !-Box:

\[
\left[ \phi_{[\hat{a}]_{B}} \right]^{B} = \]

C2: Nested !-Boxes around an edge must be nested in the same way in the rest of the tensor:

\[
\phi_{[[\hat{a}]_{A} B} \left[ \psi \right]^{A} = \]

C3: Bound edges must be in compatible !-boxes:
Conditions: C1-C3

C1: An edge entering a !-box can’t be on a node already in that !-Box:

\[
\phi_{[\hat{a}] B}^B \neq
\]

C2: Nested !-Boxes around an edge must be nested in the same way in the rest of the tensor:

\[
\phi[[\hat{a}] A B \psi]^A \neq
\]

C3: Bound edges must be in compatible !-boxes:

\[
\phi_{[\hat{a}] B} \psi_{\hat{a}} \neq
\]
Conditions: C1-C3

C1: An edge entering a !-box can’t be on a node already in that !-Box:

\[
\phi_{[\hat{a}\rangle^B}^B = \Phi_{B}^{B} \quad \text{\cancel{\begin{tikzpicture}
        
        \node at (0,0) [circle,draw=blue,fill=blue,thick] (a) {a};
        \node at (0,-1) [circle,draw=blue,fill=blue,thick] (b) {$\Phi$};
        \draw[->] (a) to (b);
      
    \end{tikzpicture}}}
\]

C2: Nested !-Boxes around an edge must be nested in the same way in the rest of the tensor:

\[
\phi_{[\hat{a}\rangle^A}_B \phi_{[\hat{a}\rangle^B}^A = \Phi_{B}^{A} \quad \text{\cancel{\begin{tikzpicture}
        
        \node at (0,0) [circle,draw=blue,fill=blue,thick] (a) {a};
        \node at (0,-1) [circle,draw=blue,fill=blue,thick] (b) {$\Phi$};
        \draw[->] (a) to (b);
        \node at (2,0) [circle,draw=blue,fill=blue,thick] (c) {B};
        \node at (2,-1) [circle,draw=blue,fill=blue,thick] (d) {\Phi};
        \draw[->] (c) to (d);
        \node at (4,0) [circle,draw=blue,fill=blue,thick] (e) {A};
        \node at (4,-1) [circle,draw=blue,fill=blue,thick] (f) {\Phi};
        \draw[->] (e) to (f);
      
    \end{tikzpicture}}}
\]

C3: Bound edges must be in compatible !-boxes:

\[
\phi_{[\hat{a}\rangle^B}^B \phi_{[\hat{a}\rangle^A}^A = \Phi_{B}^{A} \quad \text{\cancel{\begin{tikzpicture}
        
        \node at (0,0) [circle,draw=blue,fill=blue,thick] (a) {a};
        \node at (0,-1) [circle,draw=blue,fill=blue,thick] (b) {$\Phi$};
        \draw[->] (a) to (b);
        \node at (2,0) [circle,draw=blue,fill=blue,thick] (c) {B};
        \node at (2,-1) [circle,draw=blue,fill=blue,thick] (d) {\Phi};
        \draw[->] (c) to (d);
        \node at (4,0) [circle,draw=blue,fill=blue,thick] (e) {A};
        \node at (4,-1) [circle,draw=blue,fill=blue,thick] (f) {\Phi};
        \draw[->] (e) to (f);
        \node at (6,0) [circle,draw=blue,fill=blue,thick] (g) {A};
        \node at (6,-1) [circle,draw=blue,fill=blue,thick] (h) {\Phi};
        \draw[->] (g) to (h);
      
    \end{tikzpicture}}}
\]
The operation Kill removes a !-box and all nodes and edges in it:
The operation Kill removes a !-box and all nodes and edges in it: 
\[ \text{Kill}_B := \left[ [G]^B \mapsto 1, [e]^B \mapsto \epsilon, \langle e \rangle^B \mapsto \epsilon \right] \]
Operations

The operation \textit{Kill} removes a \texttt{!}-box and all nodes and edges in it: \[ \text{Kill}_B := [[G]^B \mapsto 1, [e]^B \mapsto \epsilon, \langle e \rangle^B \mapsto \epsilon] \]

Expanding a \texttt{!}-box creates a new copy of its contents with new names for all new edges/boxes. Write \( \text{fr}(G) \) for \( G \) with all names replaced by new ones (chosen by predetermined function \( \text{fr} \)).
The operation Kill removes a !-box and all nodes and edges in it: 

\[ \text{Kill}_B := [ [G]^B \mapsto 1, [e]^B \mapsto \epsilon, \langle e \rangle^B \mapsto \epsilon ] \]

Expanding a !-box creates a new copy of its contents with new names for all new edges/boxes. Write \( \text{fr}(G) \) for \( G \) with all names replaced by new ones (chosen by predetermined function \( \text{fr} \)).

\[ \text{Exp}_B := [ [G]^B \mapsto [G]^B \text{fr}(G), [e]^B \mapsto [e]^B \text{fr}(e), \langle e \rangle^B \mapsto \text{fr}(e)\langle e \rangle^B ] \]
We would like to do induction on !-box $B$ splitting into a base case (after $\text{Kill } B$) and an inductive step (proving $\text{Exp } B$ from original).
We would like to do induction on $!$-box $B$ splitting into a base case (after $\text{Kill} B$) and an inductive step (proving $\text{Exp} B$ from original).
We would like to do induction on \( !\text{-box } B \) splitting into a base case (after \( Kill_B \)) and an inductive step (proving \( Exp_B \) from original).
Induction

\[
\frac{\text{Kill}_B(G = H)}{(G = H) \implies \text{Exp}_B(G = H)} \implies \quad G = H \quad \text{(Induction)}
\]
Induction

\[ \frac{\text{Kill}_B(G = H) \quad \text{Fix}_B(G = H)}{\text{Exp}_B(G = H)} \quad \Rightarrow \quad G = H \]  

(Induction)
Spider Theorem ($\text{Kill}_B$)

Theorem

\[
A = A = A
\]

(base)
Spider Theorem \((\text{Kill}_B)\)

Theorem

\[
\begin{align*}
A & = (\text{base}) \\
A & = A
\end{align*}
\]

Proof.
Spider Theorem ($\text{Kill}_B$)

**Theorem**

**Proof.**
Spider Theorem \((\text{Fix}_B \implies \text{Exp}_B)\)

Theorem

\[
\begin{array}{c}
\text{(step)} \\
\end{array}
\]
Spider Theorem \((\text{Fix}_B \implies \text{Exp}_B)\)

**Theorem**

\[
\begin{align*}
A &\quad B \\
\text{IH} &\quad A
\end{align*}
\]

**Proof.**

\[
\begin{align*}
A &\quad B \\
\text{IH} &\quad A
\end{align*}
\]
Spider Theorem \((\text{Fix}_B \implies \text{Exp}_B)\)

**Theorem**

\[
\begin{align*}
\text{A} & \quad \text{B} \\
\text{A} & \quad \text{B}
\end{align*}
\]

Proof.

\[
\begin{align*}
\text{A} & \quad \text{B} \\
\text{A} & \quad \text{B}
\end{align*}
\]

(\text{step})
Spider Theorem \((\text{Fix}_B \implies \text{Exp}_B)\)

**Theorem**

\[
\begin{align*}
\text{A} & \quad \text{B} \\
\text{A} & \quad \text{B} \\
\text{A} & \quad \text{B} \\
\end{align*}
\]

\[
\Rightarrow
\]

\[
\begin{align*}
\text{A} & \quad \text{B} \\
\text{A} & \quad \text{B} \\
\text{A} & \quad \text{B} \\
\end{align*}
\]

(\text{step})

**Proof.**

\[
\begin{align*}
\text{A} & \quad \text{B} \\
\text{A} & \quad \text{B} \\
\text{A} & \quad \text{B} \\
\end{align*}
\]

\[
\Rightarrow
\]

\[
\begin{align*}
\text{A} & \quad \text{B} \\
\text{A} & \quad \text{B} \\
\text{A} & \quad \text{B} \\
\end{align*}
\]
Spider Theorem \( (\text{Fix}_B \iff \text{Exp}_B) \)

**Theorem**

\[
\begin{align*}
\text{A} & \quad \text{B} \\
\Rightarrow & \quad \Leftarrow \\
\text{A} & \quad \text{B} \\
\end{align*}
\]

\( \Rightarrow \) (step)

**Proof.**

\[
\begin{align*}
\text{A} & \quad \text{B} \\
\Rightarrow & \quad \Leftarrow \\
\text{A} & \quad \text{B} \\
\end{align*}
\]
Spider Theorem \((\text{Fix}_B \implies \text{Exp}_B)\)

Theorem

\[
\begin{align*}
\text{A} \quad \text{B} \\
\end{align*}
\]

Proof.

\[
\begin{align*}
\text{A} \quad \text{B} \\
\end{align*}
\]
Spider Theorem \((\text{Fix}_B \implies \text{Exp}_B)\)

**Theorem**

\[ A \quad B \quad A \quad B = \quad \text{(step)} \]

**Proof.**

\[ A \quad B \quad A \quad B = \quad \text{IH} \quad \text{IH} \quad A \quad B \quad A \quad B \]
Anti-Homomorphism
Anti-Homomorphism
Anti-Homomorphism
Table of Contents

Introduction

Tensor notation

Definitions

Induction

Summary
Summary
Summary

- Diagrammatic language:
Summary

- Diagrammatic language:

- Tensor notation:

\[
\left[ \psi^f_\hat{a} \hat{b} \right] \Phi^B_{\langle \hat{a} \rangle} [\hat{b}]^B \hat{c} \hat{d} \hat{e}
\]
Summary

- Diagrammatic language:

- Tensor notation:

- Recursive definitions:
Summary

• Diagrammatic language:

• Tensor notation:

• Recursive definitions:

• !-Box Induction: