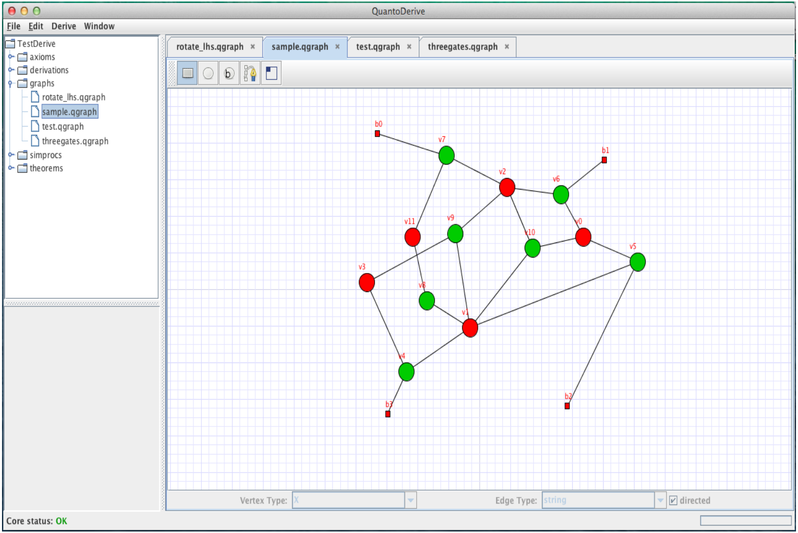
Tensors, !-graphs, and non-commutative quantum structures

Aleks Kissinger David Quick

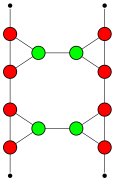
QPL
June 2014

Table of Contents

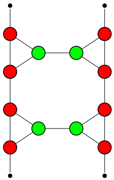
Quanto

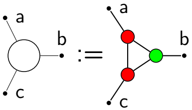

QuantoDerive
File Edit Derive Window
TestDerive
axioms
derivations
graphs
rotate_lhs.qgraph
sample.qgraph
test.qgraph
threegates.qgraph
simprocs
theorems
rotate_lhs.qgraph
sample.qgraph
test.qgraph
threegates.qgraph
b0
v7
b1
v2
v6
v11
v9
v3
v5
b2
Vertex Type:
Edge Type:
string
directed
Core status: OK

Defining Nodes

Given the graph: 

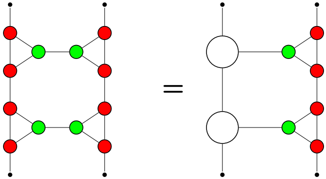
Defining Nodes

Given the graph: 

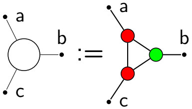
Could we define: 

Defining Nodes

Given the graph:

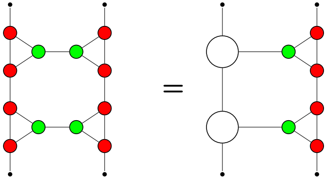


Could we define:

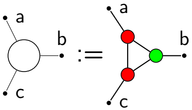


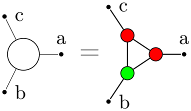
Defining Nodes

Given the graph:



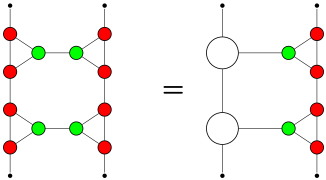
Could we define:



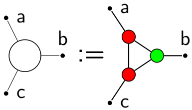


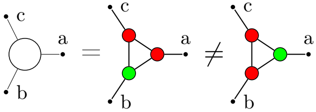
Defining Nodes

Given the graph:



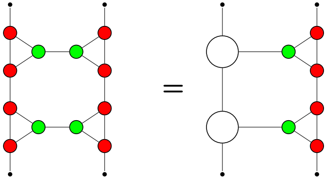
Could we define:



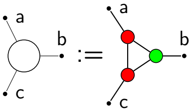


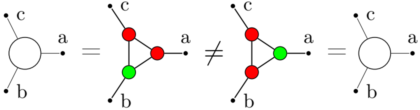
Defining Nodes

Given the graph:



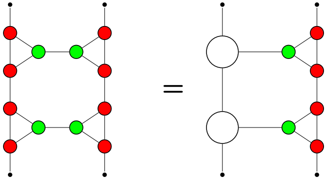
Could we define:



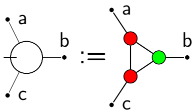


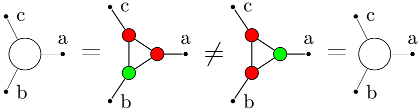
Defining Nodes

Given the graph:



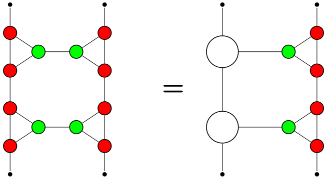
Could we define:



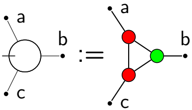


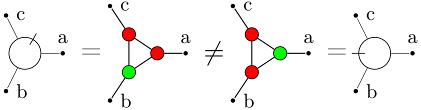
Defining Nodes

Given the graph:



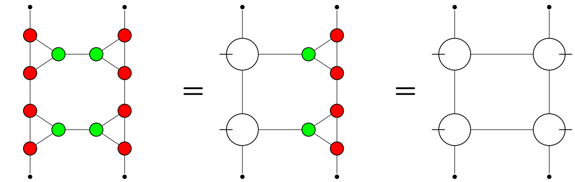
Could we define:



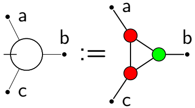


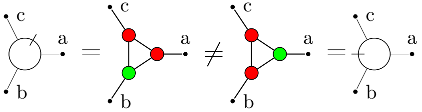
Defining Nodes

Given the graph:

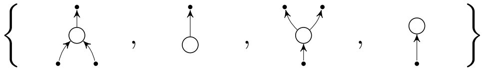


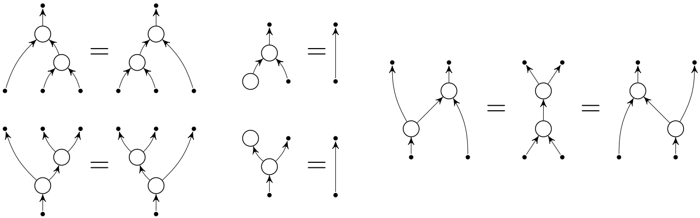
Could we define:



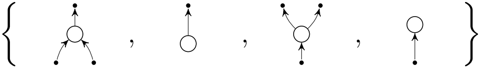


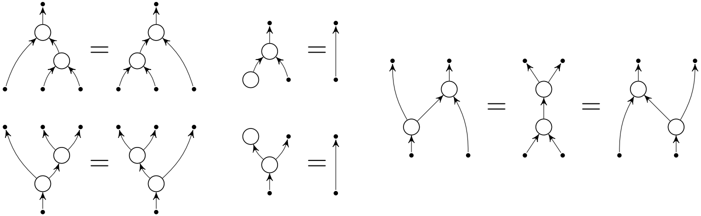
Recursively Defining Nodes

Given $\Sigma =$  satisfying:

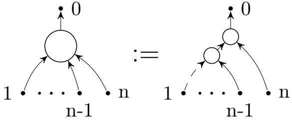


Recursively Defining Nodes

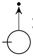
Given $\Sigma =$  satisfying:



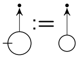
We may want to define nodes of the form:

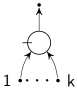


Recursively Defining Nodes

0 inputs: := 

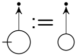
Recursively Defining Nodes

0 inputs: 

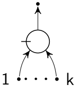
Given k inputs: 

Recursively Defining Nodes

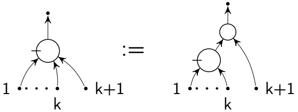
0 inputs:



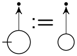
Given k inputs:

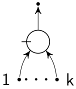


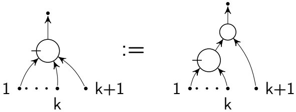
define k+1 inputs:



Recursively Defining Nodes

0 inputs: 

Given k inputs: 

define k+1 inputs: 

Recursive definitions suggest proof by induction. To do this we need to be more formal with variable arity nodes.

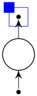
!-Boxes

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

!-Boxes

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

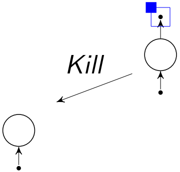
Then we have operations allowing deletion of !-boxes and creation of a new instance of the contents:



!-Boxes

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

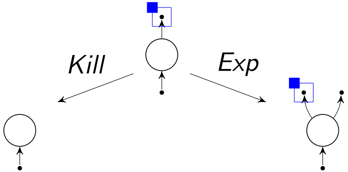
Then we have operations allowing deletion of !-boxes and creation of a new instance of the contents:



!-Boxes

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

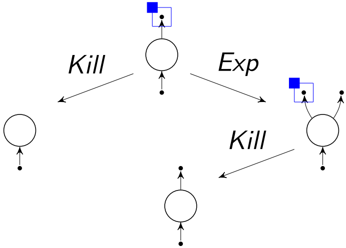
Then we have operations allowing deletion of !-boxes and creation of a new instance of the contents:



!-Boxes

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

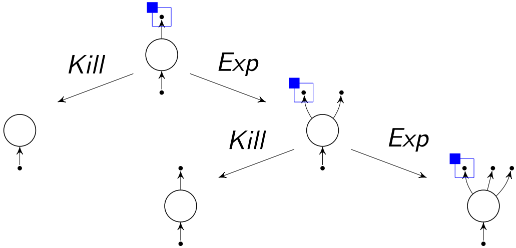
Then we have operations allowing deletion of !-boxes and creation of a new instance of the contents:



!-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:

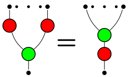


!-Box Equations

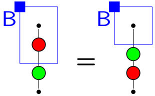
!-Boxes can be used in equations:

!-Box Equations

!-Boxes can be used in equations:

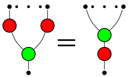


is replaced by

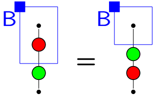


!-Box Equations

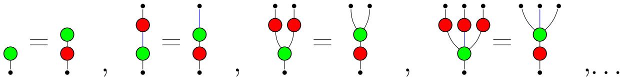
!-Boxes can be used in equations:



is replaced by

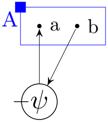


Which represents concrete equations like:



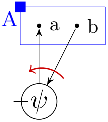
!-Box Expansion

This !-box (labelled A) has many different possible expansions:



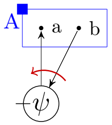
!-Box Expansion

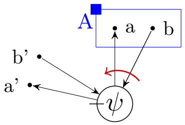
This !-box (labelled A) has many different possible expansions:



!-Box Expansion

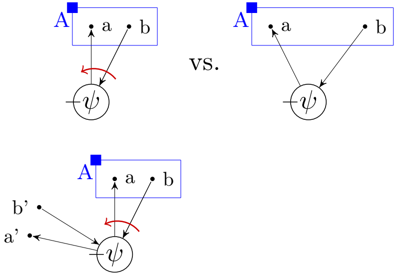
This !-box (labelled A) has many different possible expansions:





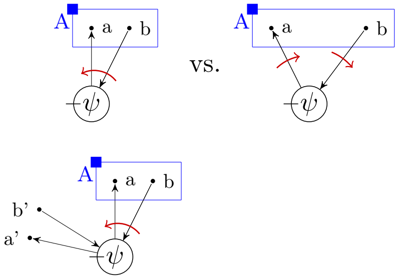
!-Box Expansion

This !-box (labelled A) has many different possible expansions:



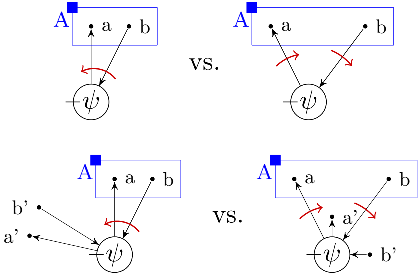
!-Box Expansion

This !-box (labelled A) has many different possible expansions:



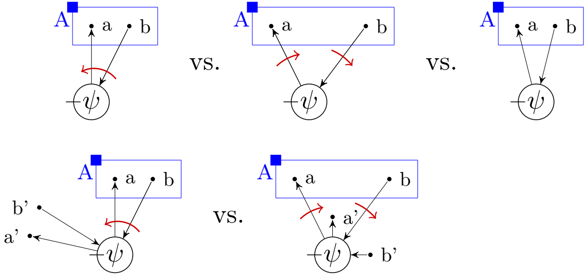
!-Box Expansion

This !-box (labelled A) has many different possible expansions:



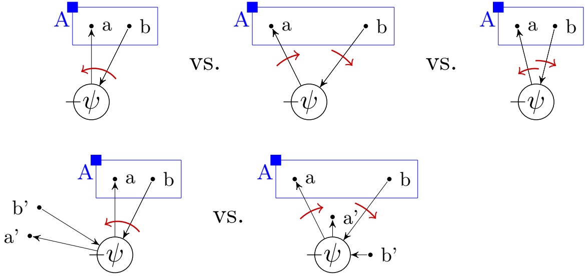
!-Box Expansion

This !-box (labelled A) has many different possible expansions:



!-Box Expansion

This !-box (labelled A) has many different possible expansions:



!-Box Expansion

This !-box (labelled A) has many different possible expansions:

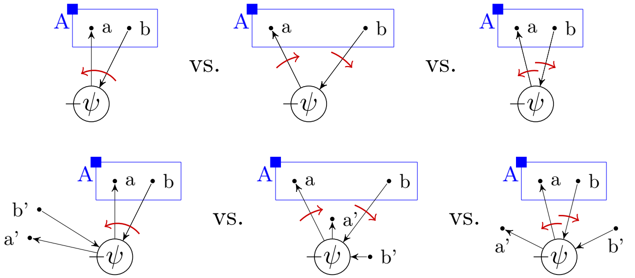
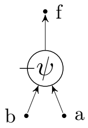


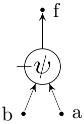
Table of Contents

Building a Tensor



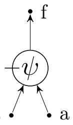
Building a Tensor

ψ =

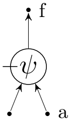


Building a Tensor

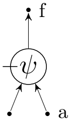
ψ_{fab} =

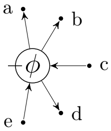


Building a Tensor

$\psi_{\hat{f}\breve{a}\check{b}} =$ 

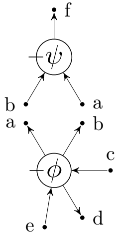
Building a Tensor

$\psi_{\hat{f}\check{a}\check{b}} =$ 

$\phi_{\hat{a}\hat{b}\check{c}\hat{d}\check{e}} =$ 

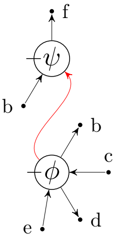
Building a Tensor

$$\psi_{\hat{f}\check{a}\check{b}}\phi_{\hat{a}\hat{b}\check{c}\hat{d}\check{e}} \quad = $$



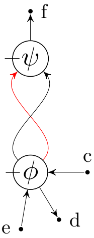
Building a Tensor

$$\psi_{\hat{f}\check{a}\check{b}}\phi_{\hat{a}\hat{b}\check{c}\hat{d}\check{e}} =$$



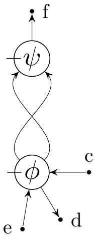
Building a Tensor

$$\psi_{\hat{f}\check{a}\check{b}}\phi_{\hat{a}\hat{b}\check{c}\hat{d}\check{e}} \quad = $$



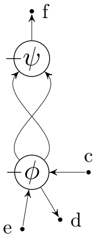
Building a Tensor

$$\psi_{\hat{f}\check{a}\check{b}}\phi_{\hat{a}\hat{b}\check{c}\hat{d}\check{e}} \quad = $$



Building a Tensor

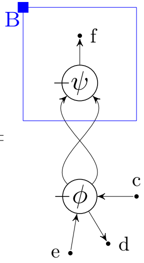
$$\psi_{\hat{f}\check{a}\check{b}}\phi_{\hat{a}\hat{b}\check{c}\hat{d}\check{e}} \quad = $$



$$= \quad \psi_{\hat{f}\check{x}\check{y}}\phi_{\hat{x}\hat{y}\check{c}\hat{d}\check{e}}$$

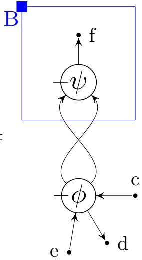
Building a Tensor

$$\psi_{\hat{f}\check{a}\check{b}}\phi_{\hat{a}\hat{b}\check{c}\hat{d}\check{e}} \quad = $$



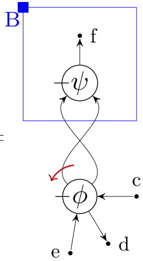
Building a Tensor

$$\left[\psi_{\hat{f}\check{a}\check{b}}\right]^{B} \phi_{\hat{a}\hat{b}\check{c}\hat{d}\check{e}} =$$



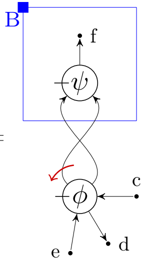
Building a Tensor

$$\left[\psi_{\hat{f}\check{a}\check{b}}\right]^{B} \phi_{\hat{a}\hat{b}\check{c}\hat{d}\check{e}} \quad =$$



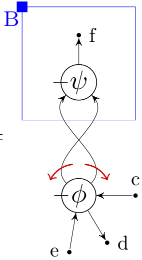
Building a Tensor

$$\left[\psi_{\hat{f}\breve{a}\breve{b}}\right]^{B}\phi_{\langle\hat{a}]^{B}\hat{b}\breve{c}\hat{d}\breve{e}} \quad =$$



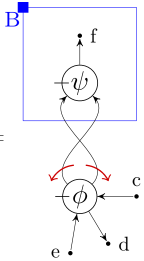
Building a Tensor

$$\left[\psi_{\hat{f}\check{a}\check{b}}\right]^{B}\phi_{\langle\hat{a}]^{B}\hat{b}\check{c}\hat{d}\check{e}} \quad =$$



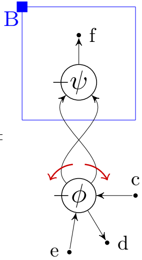
Building a Tensor

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

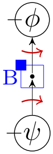


Building a Tensor

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

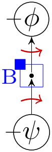


An Example



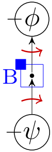
An Example

$$\psi_{[\hat{a}\rangle^B}\phi_{\langle\check{a}]^B} \quad = $$



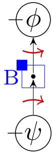
An Example

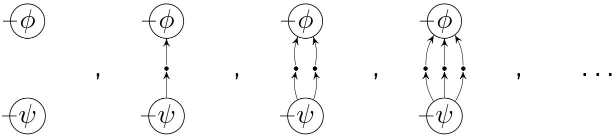
$$\psi_{[\hat{a}\rangle^B}\phi_{\langle\check{a}]^B}\left[1\right]^B \quad = $$



An Example

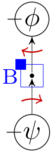
$$\psi_{[\hat{a}\rangle^B}\,\phi_{\langle\check{a}]^B}\left[1\right]^B \quad =$$





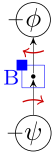
Twisted Example

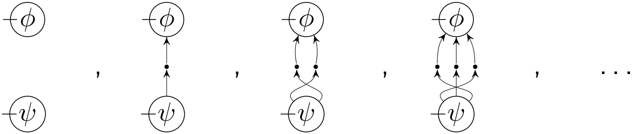
$$\psi_{[\hat{a}\rangle^B}\phi_{[\check{a}\rangle^B}\left[1\right]^B \quad = $$



Twisted Example

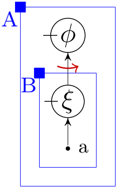
$$\psi_{[\hat{a}\rangle^B}\,\phi_{[\check{a}\rangle^B}\left[1\right]^B \quad = $$





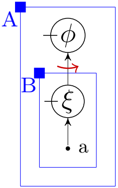
Nesting Example (Inner)

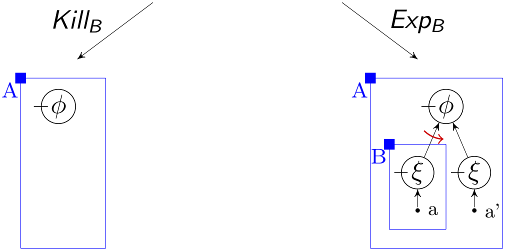
$$\left[\left[\xi_{\hat{b}\check{a}}\right]^B \phi_{\langle \check{b}]^B}\right]^A =$$



Nesting Example (Inner)

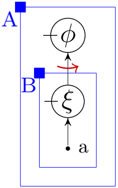
$$\left[[\xi_{\hat{b}\check{a}}]^B \phi_{\langle \check{b}]^B}\right]^A =$$





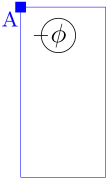
Nesting Example (Inner)

$$\left[\left[\xi_{\hat{b}\check{a}}\right]^B \phi_{\langle\check{b}]^B}\right]^A =$$

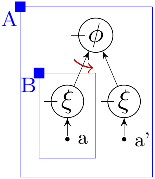


$Kill_B$

Exp_B



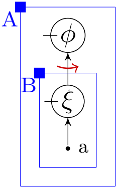
$$[\phi]^A$$



$$\left[\left[\xi_{\hat{b}\check{a}}\right]^B \xi_{\hat{b}'\check{a}'} \phi_{\check{b}'\langle\check{b}]^B}\right]^A$$

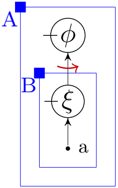
Nesting Example (Outer)

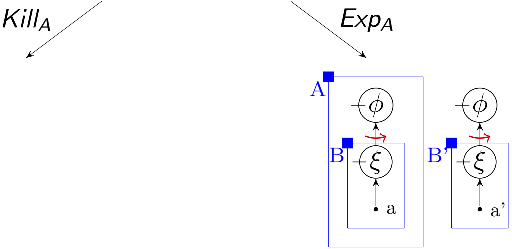
$$\left[\left[\xi_{\hat{b}\check{a}}\right]^{B}\phi_{\langle\check{b}]^{B}}\right]^{A} =$$



Nesting Example (Outer)

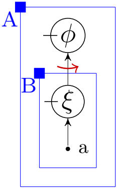
$$\left[\left[\xi_{\hat{b}\check{a}}\right]^{B} \phi_{\langle \check{b}]^{B}}\right]^{A} =$$





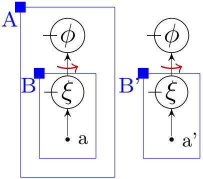
Nesting Example (Outer)

$$\left[\left[\xi_{\hat{b}\check{a}}\right]^{B} \phi_{\langle \check{b}]^{B}}\right]^{A} =$$



$Kill_A$ $\qquad$ Exp_A

1



$$\left[\left[\xi_{\hat{b}\check{a}}\right]^{B} \phi_{\langle \check{b}]^{B}}\right]^{A} \left[\xi_{\hat{b}'\check{a}'}\right]^{B'} \phi_{\langle \check{b}']^{B'}}$$

Table of Contents

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

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

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

$$\epsilon e \equiv e \equiv e\epsilon \qquad e(fg) \equiv (ef)g \qquad [\epsilon\rangle^A \equiv \epsilon \equiv \langle\epsilon]^A$$

!-Tensors

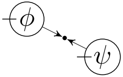
Definition

The set of all !-tensor expressions $\mathcal{T}_\Sigma$ for a signature Σ is defined recursively as:

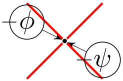
- $1 \in \mathcal{T}_\Sigma$ (empty tensor)
- $1_{\hat{a}\check{b}} \in \mathcal{T}_\Sigma$ $a, b \in \mathcal{E}$
- $\phi_e \in \mathcal{T}_\Sigma$ $e \in \mathcal{T}_e, \phi \in \Sigma$
- $\left[G\right]^A \in \mathcal{T}_\Sigma$ $G \in \mathcal{T}_\Sigma,\ A \in \mathcal{B}$
- $GH \in \mathcal{T}_\Sigma$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

Satisfying conditions F1-2, C1-3

Conditions: F1-F2

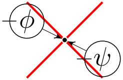
F1: No directed edge name can appear more than once as these can not be plugged together: $\phi_{\hat{a}}\psi_{\hat{a}} \quad =$ 

Conditions: F1-F2

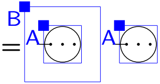
F1: No directed edge name can appear more than once as these can not be plugged together: $\phi_{\hat{a}}\psi_{\hat{a}} \quad =$ 

Conditions: F1-F2

F1: No directed edge name can appear more than once as these can not be plugged together: $\phi_{\hat{a}}\psi_{\hat{a}} \quad =$

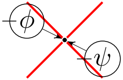


F2: No box can appear more than once to prevent overlap (which we do not allow in this formalism): $[[\ldots]^{A}]^{B}[\ldots]^{A} \quad =$

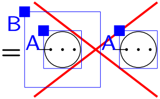


Conditions: F1-F2

F1: No directed edge name can appear more than once as these can not be plugged together: $\phi_{\hat{a}}\psi_{\hat{a}} \quad =$

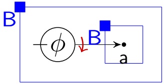


F2: No box can appear more than once to prevent overlap (which we do not allow in this formalism): $[[\dots]^A]^B[\dots]^A \quad =$



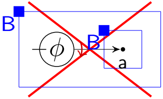
Conditions: C1-C3

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

!-Box: $\left[\phi_{[\hat{a}\rangle^{B}}\right]^{B} \quad =$ 

Conditions: C1-C3

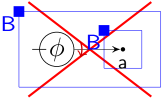
C1: An edge entering a !-box can't be on a node already in that

!-Box: $\left[\phi_{[\hat{a}\rangle^B}\right]^B \quad =$ 

Conditions: C1-C3

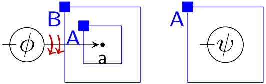
C1: An edge entering a !-box can't be on a node already in that

!-Box: $\left[\phi_{[\hat{a}\rangle^B}\right]^B \quad =$



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

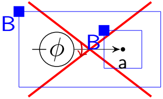
in the rest of the tensor: $\phi_{[[\hat{a}\rangle^A\rangle^B}\left[\psi\right]^A \quad =$



Conditions: C1-C3

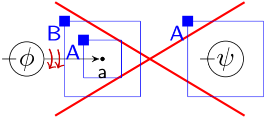
C1: An edge entering a !-box can't be on a node already in that

!-Box: $\left[\phi_{[\hat{a}\rangle^B}\right]^B \quad =$



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

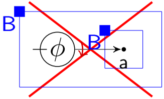
in the rest of the tensor: $\phi_{[[\hat{a}\rangle^A\rangle^B}\left[\psi\right]^A \quad =$



Conditions: C1-C3

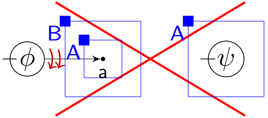
C1: An edge entering a !-box can't be on a node already in that

!-Box: $\left[\phi_{[\hat{a}\rangle^B}\right]^B \quad =$



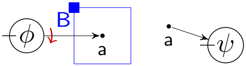
C2: Nested !-Boxes around an edge must be nested in the same way

in the rest of the tensor: $\phi_{[[\hat{a}\rangle^A\rangle^B}\left[\psi\right]^A \quad =$



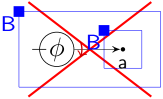
C3: Bound edges must be in compatible !-boxes:

$\phi_{[\hat{a}\rangle^B}\psi_{\check{a}} \quad =$

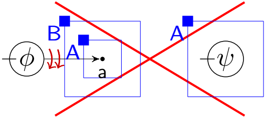


Conditions: C1-C3

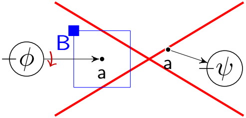
C1: An edge entering a !-box can't be on a node already in that

!-Box: $\left[\phi_{[\hat{a}\rangle^B}\right]^B \quad =$ 

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

in the rest of the tensor: $\phi_{[[\hat{a}\rangle^A\rangle^B}\left[\psi\right]^A \quad =$ 

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

$\phi_{[\hat{a}\rangle^B}\psi_{\check{a}} \quad =$ 

Operations

The operation Kill removes a !-box and all nodes and edges in it:

Operations

The operation Kill removes a !-box and all nodes and edges in it:
$Kill_B := [\left[G\right]^B \mapsto 1, [e\rangle^B \mapsto \epsilon, \langle e]^B \mapsto \epsilon]$

Operations

The operation Kill removes a !-box and all nodes and edges in it:
$Kill_B := [\big[G\big]^B \mapsto 1, [e\rangle^B \mapsto \epsilon, \langle e]^B \mapsto \epsilon]$

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

Operations

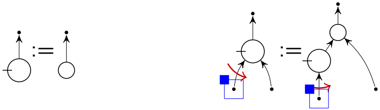
The operation Kill removes a !-box and all nodes and edges in it:
$Kill_B := [\left[G\right]^B \mapsto 1, [e\rangle^B \mapsto \epsilon, \langle e]^B \mapsto \epsilon]$

Expanding a !-box creates a new copy of its contents with new names for all new edges/boxes. Write $\mathbf{fr}(G)$ for G with all names replaced by new ones (choosen by predetermined function **fr**).
$Exp_B := [\left[G\right]^B \mapsto \left[G\right]^B \mathbf{fr}(G), [e\rangle^B \mapsto [e\rangle^B \mathbf{fr}(e), \langle e]^B \mapsto \mathbf{fr}(e)\langle e]^B]$

Table of Contents

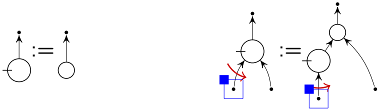
Spider Node

Definition

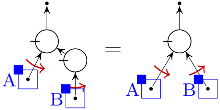


Spider Node

Definition

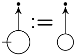


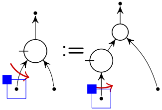
Theorem



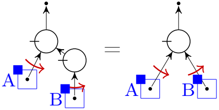
Spider Node

Definition





Theorem



We would like to do induction on !-box B splitting into a base case (after $Kill_B$) and an inductive step (proving Exp_B from original).

Induction

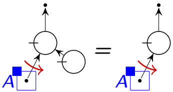
$$\frac{Kill_B(G = H) \qquad\qquad (G = H) \implies Exp_B(G = H)}{G = H} \text{ (Induction)}$$

Induction

$$\frac{Kill_B(G = H) \qquad Fix_B(G = H) \implies Exp_B(G = H)}{G = H} \text{ (Induction)}$$

Spider Theorem ($Kill_B$)

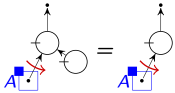
Theorem



(base)

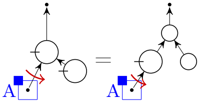
Spider Theorem ($Kill_B$)

Theorem



(base)

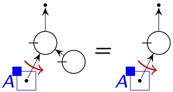
Proof.



□

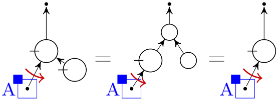
Spider Theorem ($Kill_B$)

Theorem



(base)

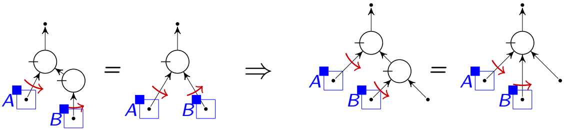
Proof.



□

Spider Theorem ($Fix_B \implies Exp_B$)

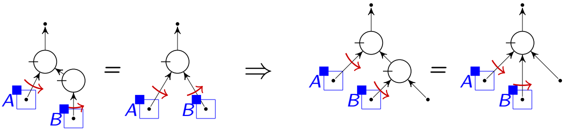
Theorem



(step)

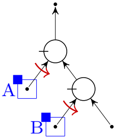
Spider Theorem ($Fix_B \implies Exp_B$)

Theorem



(step)

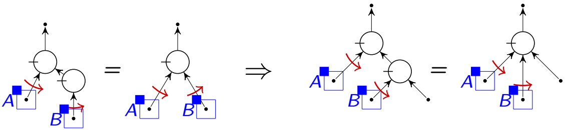
Proof.



□

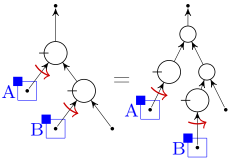
Spider Theorem ($Fix_B \implies Exp_B$)

Theorem



(step)

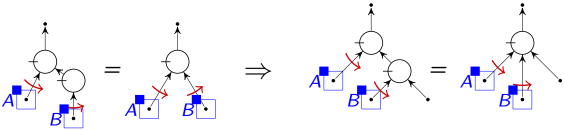
Proof.



□

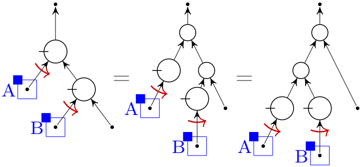
Spider Theorem ($Fix_B \implies Exp_B$)

Theorem



(step)

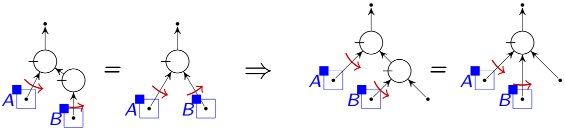
Proof.



□

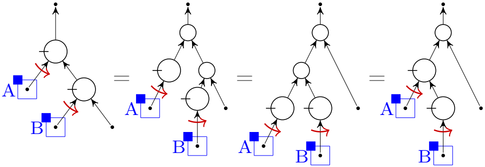
Spider Theorem ($Fix_B \implies Exp_B$)

Theorem



(step)

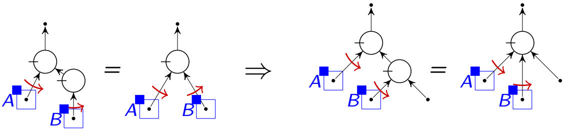
Proof.



□

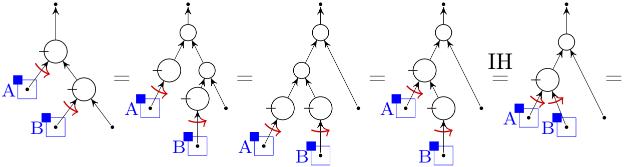
Spider Theorem ($Fix_B \implies Exp_B$)

Theorem



(step)

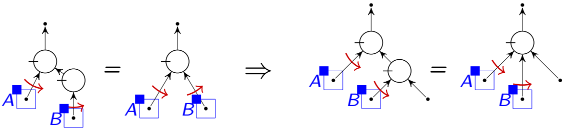
Proof.



□

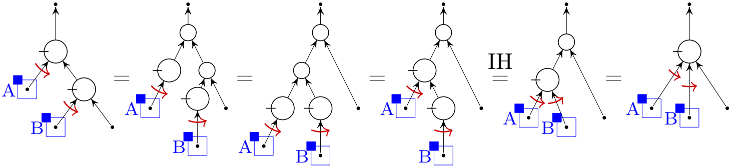
Spider Theorem ($Fix_B \implies Exp_B$)

Theorem



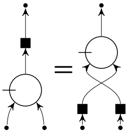
(step)

Proof.

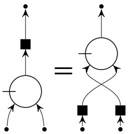


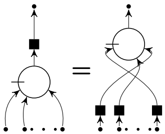
□

Anti-Homomorphism



Anti-Homomorphism





Anti-Homomorphism

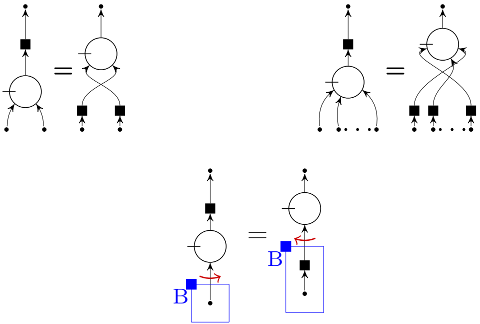
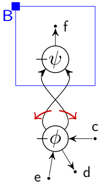


Table of Contents

Summary

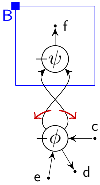
Summary

- Diagrammatic language:



Summary

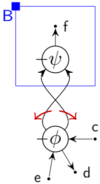
- Diagrammatic language:



- Tensor notation: $\left[\psi_{\hat{f}\check{a}\check{b}}\right]^{B}\phi_{\langle\hat{a}]^{B}[\hat{b}\rangle^{B}\check{c}\hat{d}\check{e}}$

Summary

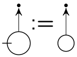
- Diagrammatic language:

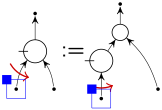


- Tensor notation:

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

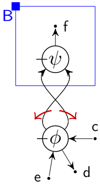
- Recursive definitions:





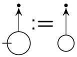
Summary

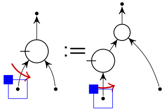
- Diagrammatic language:



- Tensor notation: $\left[\psi_{\hat{f}\check{a}\check{b}}\right]^{B} \phi_{\langle\hat{a}]^{B}[\hat{b}\rangle^{B}\check{c}\hat{d}\check{e}}$

- Recursive definitions:





- !-Box Induction:

$$\frac{Kill_B(G = H) \qquad Fix_B(G = H) \implies Exp_B(G = H)}{G = H} \text{ (Induction)}$$