# Tensor networks (ETH 09.09.2019)

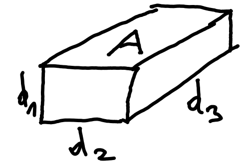
by Maris Ozols

## Outline:

1. TN formalism
2. TNs and open quantum systems
3. Quantum circuits as TNs

## 1. Tensor network formalism

Tensor = a brick of numbers (an array)




Can think of it as a function

$$A: \underset{i \in}{\{1, \dots, d_1\}} \times \underset{j \in}{\{1, \dots, d_2\}} \times \underset{k \in}{\{1, \dots, d_3\}} \to \mathbb{C}$$

Tensor entries:

$A[i, j, k]$ ← for programmers

$A_{ijk}$ ← for this talk

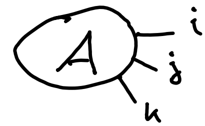
The set of all (complex) tensors of a given shape:

$$T(d_1, d_2, d_3)$$

Mathematically this is the same as a vector in a tensor product space:

$$\mathbb{C}^{d_1} \otimes \mathbb{C}^{d_2} \otimes \mathbb{C}^{d_3}$$

## Graphical notation:




Examples:

scalar   vector   matrix


$\in T$   $\in T(d)$   $\in T(d_1, d_2)$   bond dimensions

## Addition:

If $A, B \in T(d_1, d_2)$ then $A + B = C \in T(d_1, d_2)$ such that

$$A_{ij} + B_{ij} = C_{ij} \quad \forall i, j$$

## Standard basis:

$E(i,j) \in T(d_1, d_2)$

s.t. $E(i,j)_{i'j'} = \delta_{ii'} \delta_{jj'}$   $E(i,j) = \begin{array}{cc} & j \\ i & \begin{pmatrix} \cdots & 1 & \\ & 0 & \end{pmatrix} \end{array}$

E.g., $E(i) = (0, \dots, 0, \overset{i}{1}, 0, \dots, 0)$ and

## Tensor product:

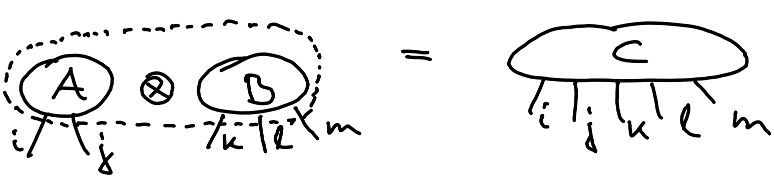
If $A \in T(d_1, d_2)$ and $B \in T(d_3, d_4, d_5)$ then

$$A \otimes B = C \in T(d_1, \dots, d_5)$$

such that

$$A_{ij} \cdot B_{klm} = C_{ijklm}$$

Graphical notation:



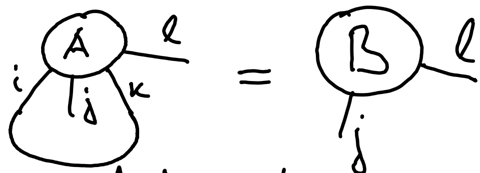
Standard basis elements compose nicely under tensor product:

$$E(i,j) \otimes E(k,l,m) = E(i,j,k,l,m)$$

entries given by a product of five delta functions

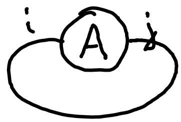
Not every tensor is a tensor product of two tensors. E.g., $E(1)\otimes E(1) + E(2)\otimes E(2)$ is not.

<u>Contraction</u>:



If $A \in T(d_1,d_2,d_3,d_4)$ with $d_1 = d_3$ then contracting indices 1 and 3 of $A$ produces another tensor $B \in T(d_2,d_4)$ s.t.

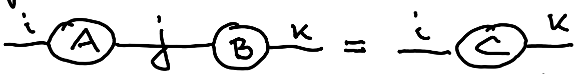
$$B_{jl} = \sum_{i=1}^{d_1} A_{ijil}$$

Example:  $= \sum_{i=1}^{d} A_{ii} = \mathrm{Tr}\, A$

Same idea works for a pair of tensors because you can treat their tensor product as a single tensor. E.g.,

 where $w_i = \sum_{j=1}^{d} A_{ij} v_j$

This corresponds to matrix-vector product: $Av = w$. Similarly,

 where $C_{ik} = \sum_{j=1}^{d} A_{ij} B_{jk}$

corresponds to matrix-matrix product:

$$A \cdot B = C.$$

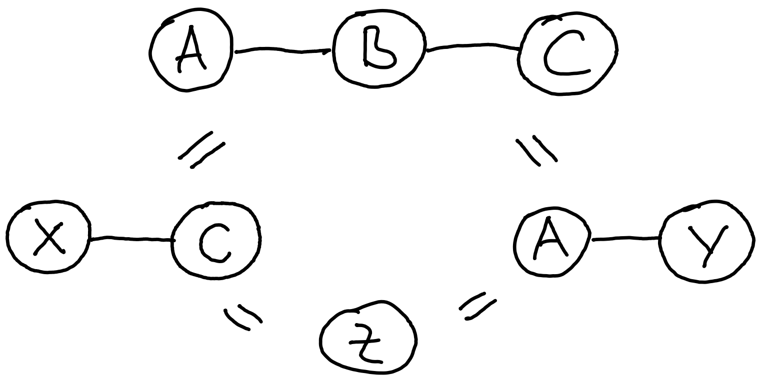
<u>Associativity</u>: matrix multiplication is associative, i.e.,

$$(AB)C = A(BC)$$

since

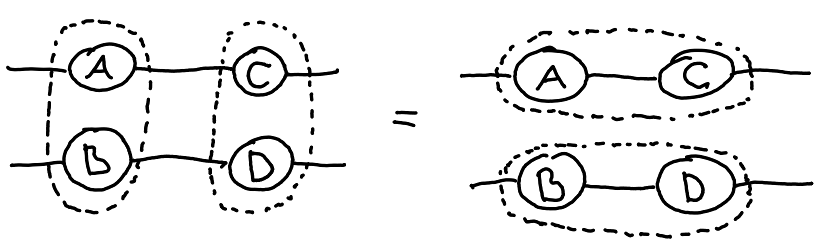
$$\sum_k \Big( \sum_j A_{ij} B_{jk} \Big) C_{kl} = \sum_j A_{ij} \Big( \sum_k B_{jk} C_{kl} \Big).$$

Thus we can unambiguously write $ABC$. Same goes for tensor contraction - when multiple edges must be contracted to evaluate an expression, it does not matter in what order we do it.

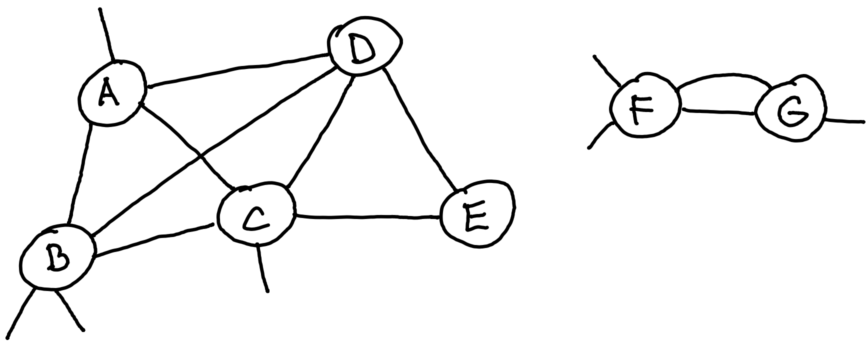


Contraction is also compatible with the tensor product in the following sense:

$$(A \otimes B) \cdot (C \otimes D) = (A \cdot C) \otimes (B \cdot D)$$



<u>Tensor network</u> is a collection of tensors, with various pairs of indices being contracted.
It can be naturally represented by a graph (possibly with some dangling edges):




The value of a general tensor network with graph $(V,E)$, no dangling edges and all indices in $[d]$ is

$$\boxed{\sum_{s\in[d]^E}\ \prod_{v\in V} A(v)_{s(v)}}$$

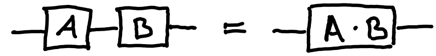
where $[d]^E = \{E \to \{1,\dots,d\}\}$
= the set of all functions from $E$ to the set $[d]$
= a vector with entries in $[d]$, indexed by $E$

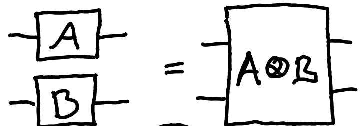
$s(v)$ = $s$ restricted to edges incident to $v$

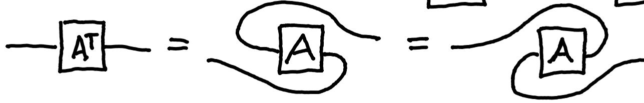
$A(v)_{s(v)}$ = the entry of tensor $A(v)$ when its indices are restricted to $s(v)$

<u>Theorem</u>: If two tensor networks are isomorphic as vertex-labelled graphs, they yield the same tensor when contracted.

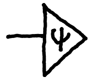
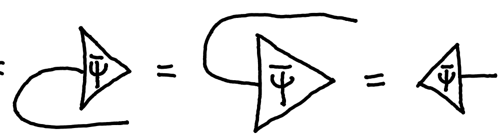
# Tensor networks and quantum information

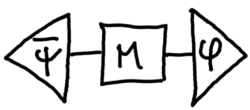
Matrix product: $A \cdot B =$ 


Tensor product: $A \otimes B =$ 

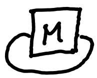
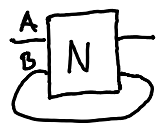

Transpose: 


Bras and kets:

$|\psi\rangle =$  $\quad \langle\psi| = |\bar{\psi}\rangle^T =$ 

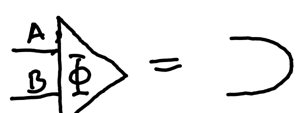

$\langle\psi|M|\varphi\rangle =$ 

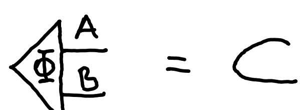

Trace and partial trace:

$\mathrm{Tr}\, M =$  and $\mathrm{Tr}_B\, N_{AB} =$ 


Identity operation: $I = \sum_{i=1}^{d} |i\rangle\langle i| =$

Un-normalized maximally entangled state:

$|\Phi\rangle_{AB} := \sum_{i=1}^{d} |i\rangle_A |i\rangle_B =$ 


$\langle\Phi|_{AB} =$ 

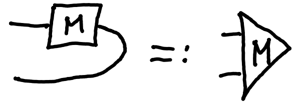

Swap:

$\mathrm{SWAP} := \sum_{i,j=1}^{d} |i\rangle \otimes |j\rangle \langle j| \otimes \langle i| =$ 

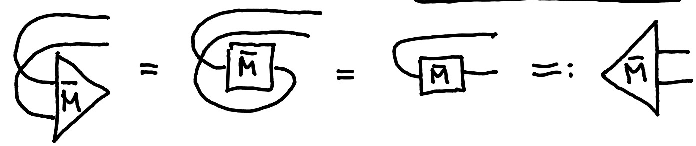
# 2. Open quantum systems

## Vectorization

If $M=\sum_{i=1}^{r}\sum_{j=1}^{c} M_{ij}\,|i\rangle\langle j|$ then $\operatorname{vec} M := \sum_{i=1}^{r}\sum_{j=1}^{c} M_{ij}\,|i\rangle\otimes|j\rangle$.

vec(M) = 

$$\boxed{\begin{aligned}&(A\otimes B)\operatorname{vec}(X)\\ &=\operatorname{vec}(AXB^T)\end{aligned}}$$

$\operatorname{vec}(M)^\dagger =$ 

## Purification

$|\psi\rangle_{AB}$ is a purification of $\rho_A$ if $\rho_A = \operatorname{Tr}_B |\psi\rangle\langle\psi|_{AB}$.
The canonical purification of $\rho$:

$$|\psi\rangle_{AB} := (\sqrt{\rho}_A\otimes I_B)\,|\Phi\rangle_{AB} = \operatorname{vec}(\sqrt{\rho}).$$

One can check that $\langle\psi|\psi\rangle = 1$ and $\operatorname{Tr}_B |\psi\rangle\langle\psi|_{AB} = \rho_A$.

## Singular value & Schmidt decomposition

Any $d_A\times d_B$ complex matrix $M$ can be written as $M = UDV$ where $U\in\mathrm{U}(d_A)$, $V\in\mathrm{U}(d_B)$ and $D$ is a diagonal $d_A\times d_B$ matrix with non-negative entries. This is known as singular value decomposition.

By vectorizing, any bipartite state $|\psi\rangle_{AB}$ can be written as

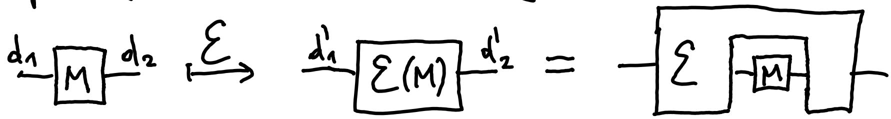
$$|\psi\rangle_{AB} = \sum_{i=1}^{r} s_i\,|u_i\rangle_A\otimes|v_i\rangle_B$$

where $s_i\geq 0$, $r:=\min\{d_A,d_B\}$ and $|u_1\rangle,\ldots,|u_r\rangle\in\mathbb{C}^{d_A}$, $|v_1\rangle,\ldots,|v_r\rangle\in\mathbb{C}^{d_B}$ are orthonormal sets of vectors.

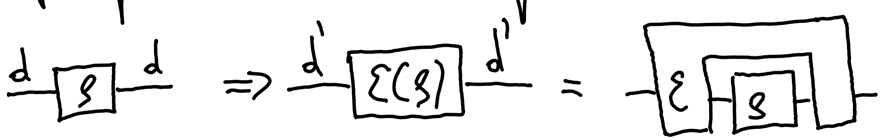
# Superoperators & quantum channels

**Superoperator** is a linear map that sends matrices to matrices.

Superoperator $\mathcal{E}$ acting on a matrix $M$:

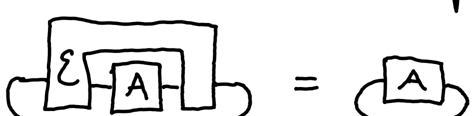



**Quantum channel** is a superoperator that maps quantum states to quantum states:




What restrictions does this put on $\mathcal{E}$?

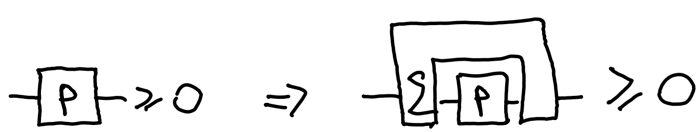
TP: trace-preserving



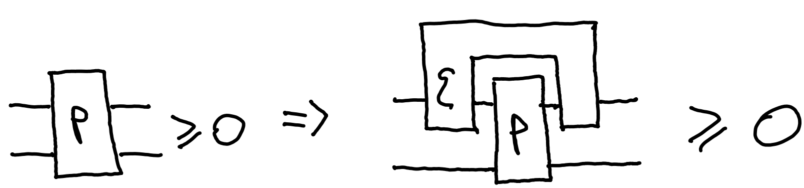

$\forall A$ (square matrix)

HP: Hermitian-preserving


PP: positivity-preserving



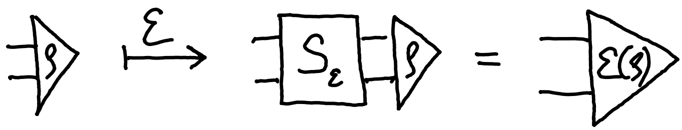

CP: completely positive



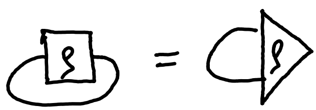
## Quantum chanels $\cong$ CP & TP superoperators

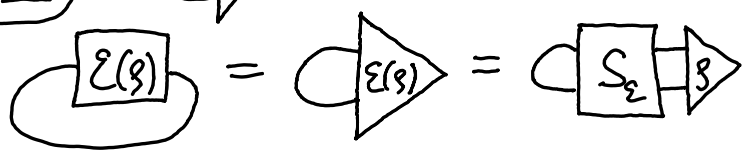
How can we represent these linear maps?

<u>The „natural" (Liouville) representation:</u>



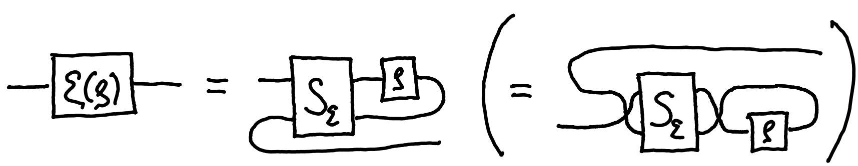

The operator $S_\mathcal{E}$ must be TP and HP:

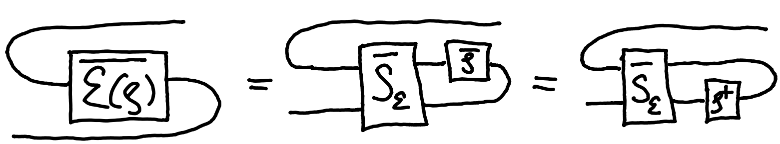
- $\mathrm{Tr}\,\rho =$ 


  $\mathrm{Tr}\,\mathcal{E}(\rho) =$ 

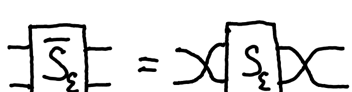

  These should agree for all $\rho$, so

  TP: 


- $\mathcal{E}(\rho) =$ 


  $\mathcal{E}(\rho)^\dagger =$ 


  Hence

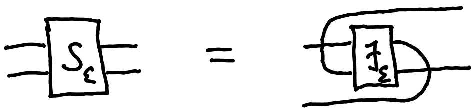
  HP: 

Capturing PP and CP is more difficult.
For this we need another representation.

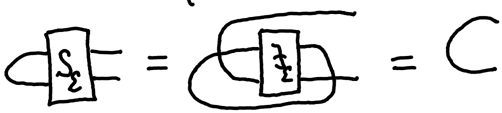
## Choi - Jamiołkowski representation

Let 

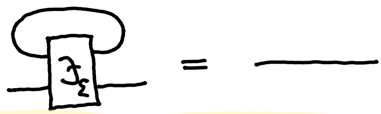
The inverse relation looks the same:



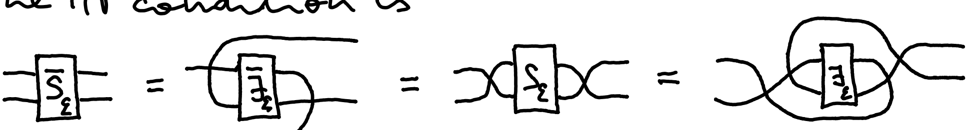
The TP condition for $J_\varepsilon$ reads as follows:

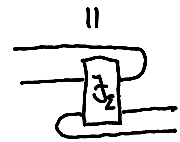


In other words:

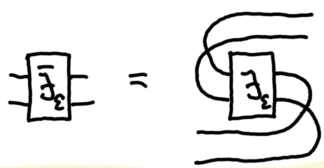
TP: 

The HP condition is





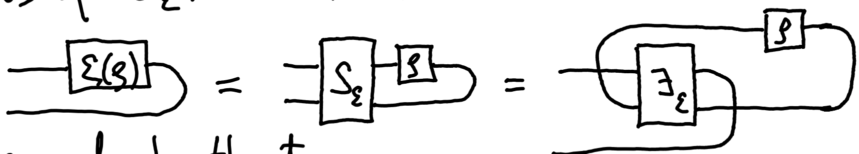
This simplifies to

HP: 

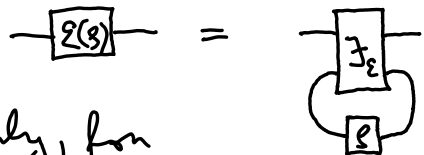
This is the same as $J_\varepsilon^\dagger = J_\varepsilon$, i.e., $J_\varepsilon$ is Hermitian!

What about the PP and CP conditions for $\mathcal{J}_\mathcal{E}$?

First, we need to express the output state $\mathcal{E}(\rho)$ in terms of $\mathcal{J}_\mathcal{E}$. Recall that

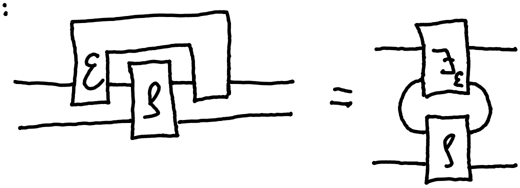



We conclude that




Similarly, for a bipartite state:

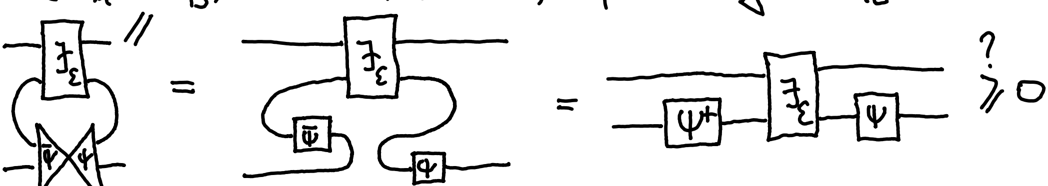
$(\mathcal{E}_A \otimes \mathcal{I}_B)(\rho_{AB}) =$



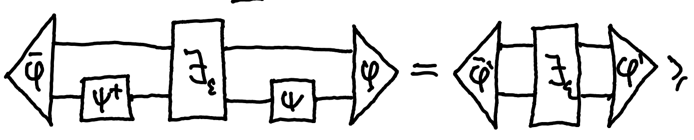

In particular, note that $(\mathcal{E}_A \otimes \mathcal{I}_B)(|\Phi\rangle\langle\Phi|_{AB}) = \mathcal{J}_\mathcal{E}$.
Hence, if $\mathcal{E}$ is CP then $\mathcal{J}_\mathcal{E} \geq 0$.

Conversely, if $\mathcal{J}_\mathcal{E} \geq 0$, let us show that $\mathcal{E}$ is CP. By linearity, it suffices to show that

$$(\mathcal{E}_A \otimes \mathcal{I}_B)(|\bar\psi\rangle\langle\bar\psi|_{AB}) \geq 0, \quad \text{for any } |\psi\rangle_{AB}.$$




Note that




$\geq 0$ since $\mathcal{J}_\mathcal{E} \geq 0$.

Hence,

CP:

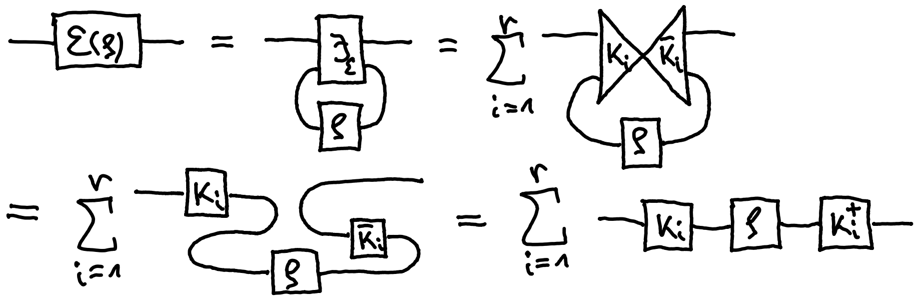

$\geq 0$

## Kraus representation

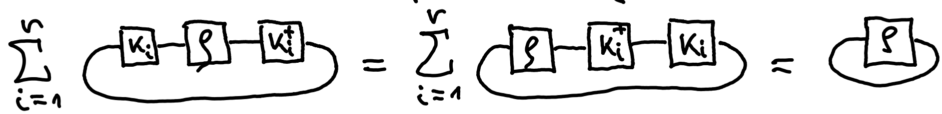
Since $J_\mathcal{E} \geq 0$, it has a spectral decomposition

$$J_\mathcal{E} = \sum_{i=1}^{r} \lambda_i \, |v_i\rangle\langle v_i| \quad \text{with } \lambda_i \geq 0 \text{ and } r = \operatorname{rank} J_\mathcal{E}.$$

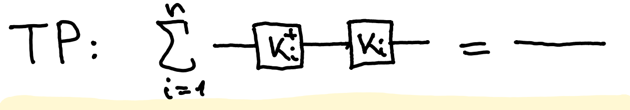
Let $|\xi_i\rangle := \sqrt{\lambda_i}\, |v_i\rangle$ so that $J_\mathcal{E} = \sum_{i=1}^{r} |K_i\rangle\langle K_i|$. Then



Since $\mathcal{E}$ is trace-preserving,

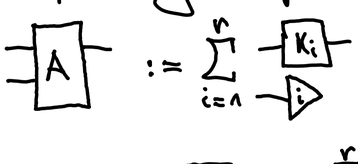


so TP:

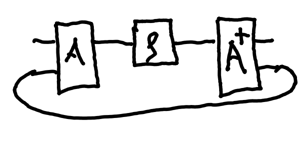


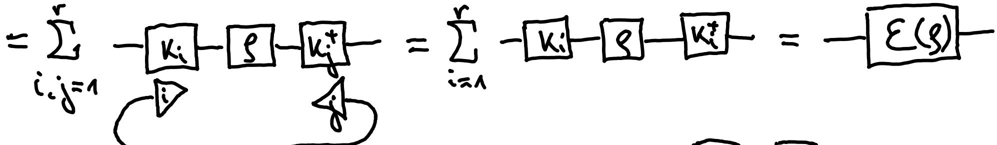
## Stinespring representation

Let

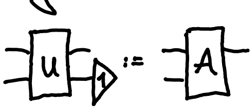


Then

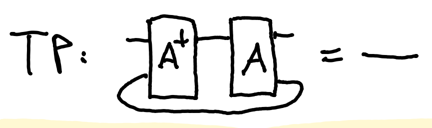




Stinespring isometry (dilation):

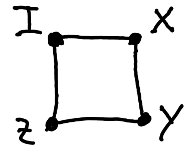


TP:



# 3. Quantum circuits as tensor networks

**Pauli matrices:**




$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

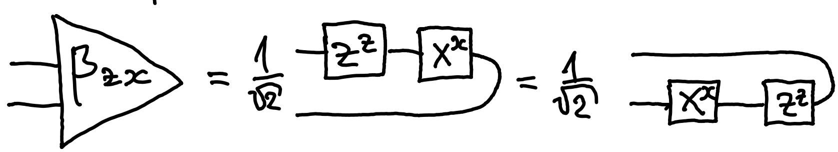
$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \qquad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} = iXZ = -iZX$$

**Bell states:**

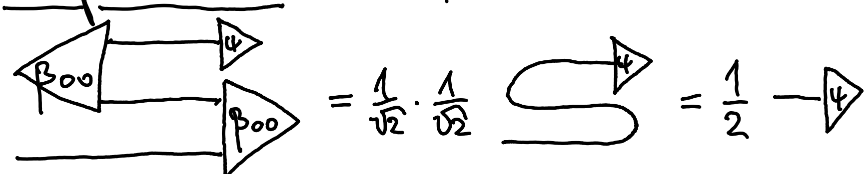
$$\sqrt{2}\,|\beta_{00}\rangle := |00\rangle + |11\rangle = \operatorname{vec} I \overset{|\Phi\rangle}{\parallel} \qquad \sqrt{2}\,|\beta_{01}\rangle = |01\rangle + |10\rangle = \operatorname{vec} X$$

$$\sqrt{2}\,|\beta_{10}\rangle := |00\rangle - |11\rangle = \operatorname{vec} Z \qquad \sqrt{2}\,|\beta_{11}\rangle = |01\rangle - |10\rangle = i \operatorname{vec} Y$$

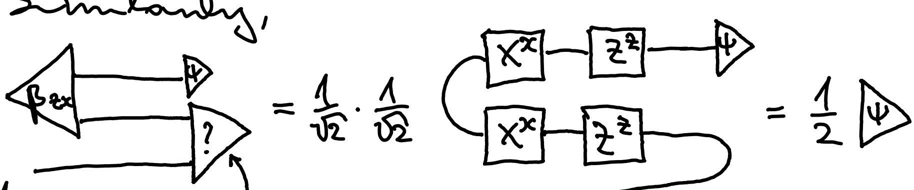
**Relation:** $|\beta_{zx}\rangle = \frac{1}{\sqrt{2}} \operatorname{vec}\left(Z^z X^x\right), \quad x, z \in \{0,1\}$



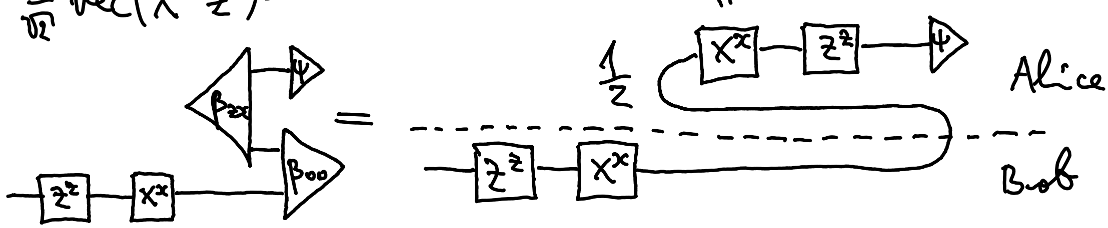

**Teleportation:** First, note that



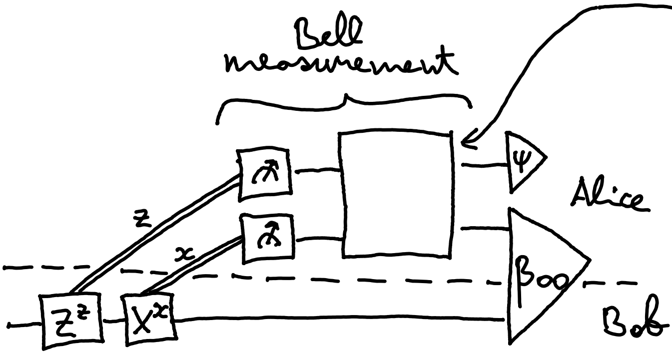

Similarly,




$\frac{1}{\sqrt{2}} \operatorname{vec}(X^x Z^z)$

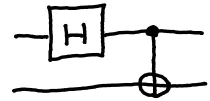


· Teleportation protocol:




$$\sum_{z,x\in\{0,1\}} |z,x\rangle\langle\beta_{zx}|$$

$$\|$$




Bob ends up with  and $z,x\in\{0,1\}$ are uniformly random. This can be generalized by taking the initial state instead of $|\psi\rangle_A$ to be, say, $|\beta_{00}\rangle_{ZA}$ where Z belongs to a third person, Zoey. The protocol ends up establishing entanglement between Zoey and Bob even though they never met.

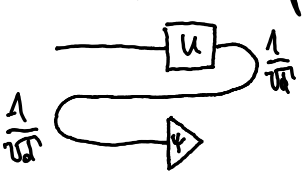
"Quantum software"

Thanks to the Choi-Jamiołkowski isomorphism, we can encode quantum operations as quantum states. But how can we execute such "quantum software"?

- Alice can encode a $d\times d$ unitary $U$ into a quantum state:

$$(U_A\otimes I_{A'})\cdot\frac{1}{\sqrt{d}}|\Phi\rangle_{AA'} = \frac{1}{\sqrt{d}}$$  $$= \frac{1}{\sqrt{d}}\,\mathrm{vec}(U)$$

To execute it on $|\psi\rangle_B$, Bob measures in the Bell basis and post-selects on $\frac{1}{\sqrt{d}}|\Phi\rangle$:

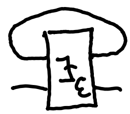


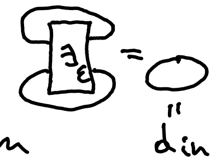

$$= \frac{1}{d}$$  Prob. $= \frac{1}{d^2}$.

If $U$ is a Clifford gate, $U Z^z X^x U^\dagger = \pm Z^{f(z)} X^{g(x)}$ so Bob can measure in the Bell basis and apply a Pauli correction to always recover the state.

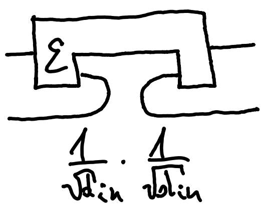
- More generally, Alice's algorithm can be a quantum channel $\mathcal{E}$ which she can encode in a state $\rho_\mathcal{E}$ by appropriately normalizing its Choi matrix $J_\mathcal{E}$:

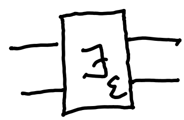
$$\rho_\mathcal{E} := c \cdot J_\mathcal{E}$$

Recall that 

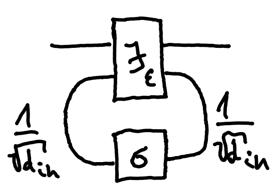
$= \overline{\quad d_{in} \quad}$ so $\mathrm{Tr}\, J_\mathcal{E} =$ 


so we need to take $c = 1/d_{in}$. Another way to see this is as follows:

$\rho_\mathcal{E} =$ 

$= \frac{1}{d_{in}}$ 


To execute $\mathcal{E}$ on $\sigma_B$, Bob measures in the Bell basis and post-selects on $\frac{1}{\sqrt{d}}|\Phi\rangle$:

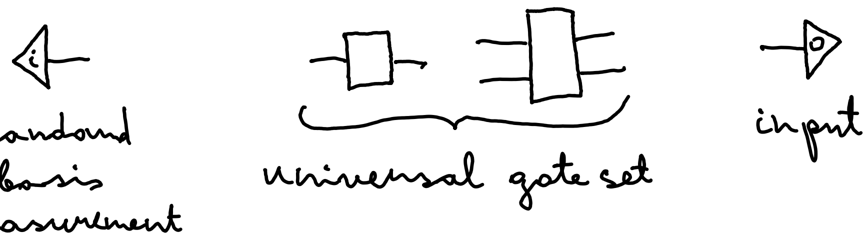
$\frac{1}{d_{in}} \cdot$ 

$= \frac{1}{d_{in}^2} \; -\boxed{\mathcal{E}(\sigma)}-$

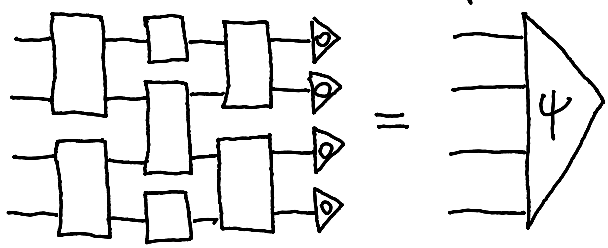
$\uparrow$ prob. of success

# Simulation of quantum circuits

Quantum circuit is naturally a tensor network. It consists of

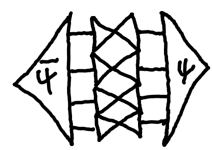



Here is an example of a quantum circuit:



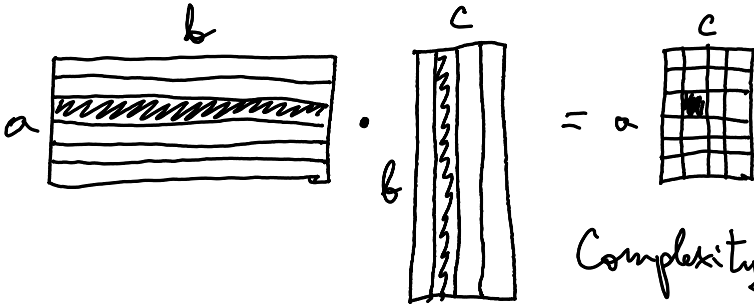

If the final $n$-qubit state $|\psi\rangle$ is measured, the probability of outcome $K \in \{0,1\}^n$ is

$$p_K = |\langle K|\psi\rangle|^2 = \langle\psi|K\rangle\langle K|\psi\rangle =$$



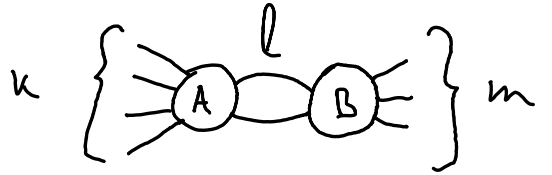

To simulate the circuit, we either want to compute $p_K$ (strong simulation) or sample from the corresponding distribution (weak simulation). Note that the value of $p_K$ can be obtained by contracting the above tensor network. This is why tensor networks provide a very useful tool for quantum simulation. But what is the compexity of this?

Complexity of matrix multiplication:




Complexity = $a \cdot b \cdot c$

Tensor contraction:

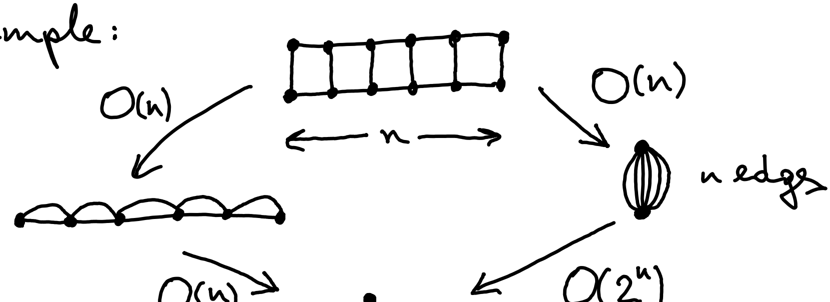



(each leg has dim. 2)

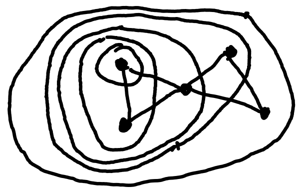
$$\text{Complexity} = 2^k \cdot 2^l \cdot 2^m = 2^{k+l+m}$$

# of free legs + # of contracted legs

Example:



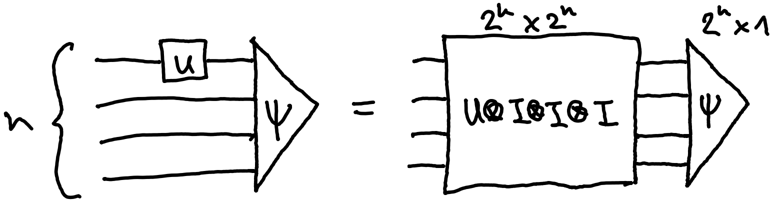

While the answer is the same, the complexity of getting it is very different. In this regard the contraction order matters! A general order can be specified by a "bubbling":



# Naive approach vs tensor networks

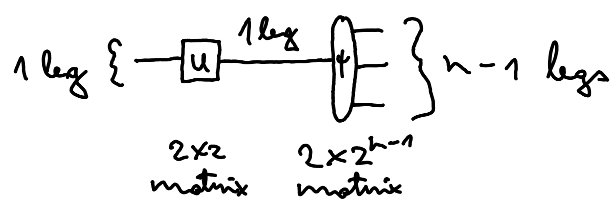
Let's say we store an $n$-qubit state $|\psi\rangle$ as a vector with $2^n$ components. How expensive is it to apply a 1-qubit gate on the first qubit?

Naive approach – treat $U$ as a $2^n \times 2^n$ matrix:




This has complexity $2^n \cdot 2^n = 2^{2n}$.

On the other hand, if we treat $U$ and $|\psi\rangle$ as tensors and only contract one index, we are effectively multiplying smaller matrices:



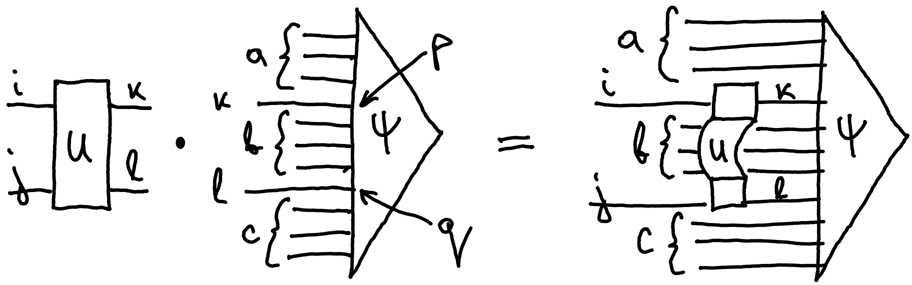

This has complexity $2 \cdot 2 \cdot 2^{n-1} = 2^{n+1}$, which is much better than the naive approach. Hence this is also the method used by many simulators, such as the one in Cirq.

Let's take a look at how to implement this in Python.

# Example Python code

If we store an $n$-qubit state $|\Psi\rangle$ as a vector with $2^n$ components, we can apply 1-qubit and 2-qubit gates by contracting the legs on which the gate acts with a tensor describing the gate itself.

Let's say we want to apply a 2-qubit gate $U$ on qubits $p<q$ of $|\Psi\rangle$:




Create all-zeroes vector with $2^n$ complex entries and initialize to $|0\rangle^{\otimes n}$:

```
state = np.zeroes(2**n, np.complex64)
state[0] = 1+0j
```

Group legs into 5 groups of sizes $p, 1, q-p-1, 1, n-q-1$:

```
psi = state.reshape([2**p, 2, 2**(q-p-1), 2, 2**(n-q-1)])
```

Contract the gate and state tensors:

```
state = np.einsum('ijkl, akblc->aibjc', U, psi)
```