Lecture 13

Prop Any CNOT+phase circuit describes a unitary of the form:

$$U :: |\vec{x}\rangle \mapsto e^{i\phi(\vec{x})} |L\vec{x}\rangle.$$

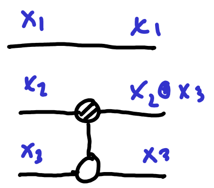
phase polynomial (ϕ)   parity matrix (L).

From the example above: $L = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$ and

$$\phi(x_1, x_2, x_3) = \underbrace{(\alpha + \gamma)}_{\text{phase-folding}} \cdot (x_2 \oplus x_3) + \beta \cdot (x_1 \oplus x_2) + \theta \cdot x_2$$

Q: can we re-synthesise a circuit for (L, ϕ)?

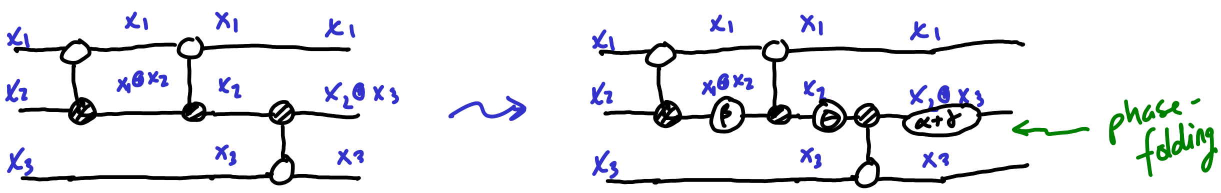
For L, we have:



To get ϕ, we need to place Z-phases on wires labelled:

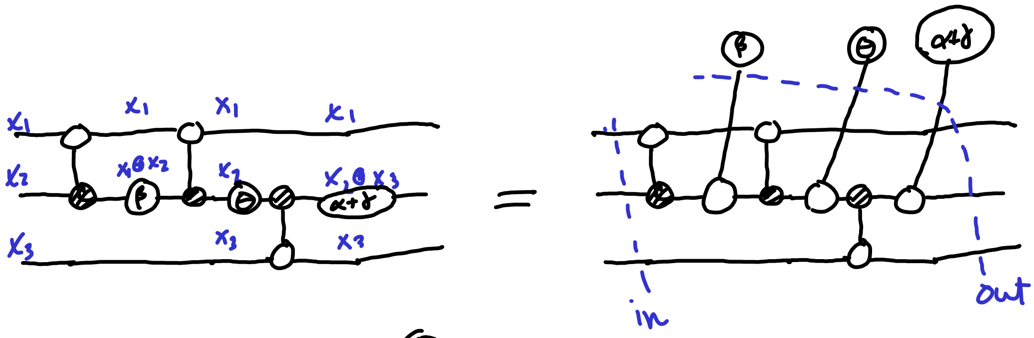
$x_2 \oplus x_3$, $x_1 \oplus x_2$, and x_2

Only $x_1 \oplus x_2$ is missing, so lets (temporarily) create it:



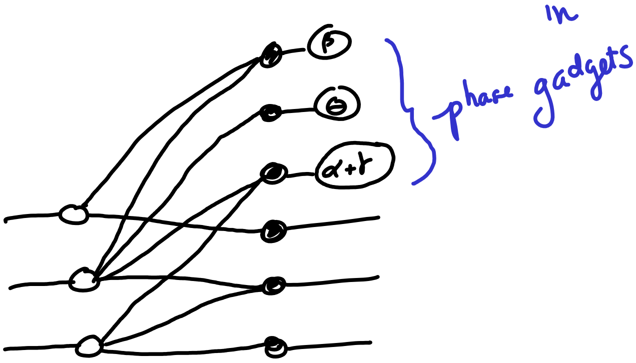
Phase polynomials, graphically (aka. phase gadgets)

Ex



phase-free simp.

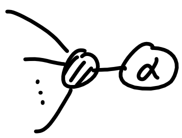
=



1-legged:

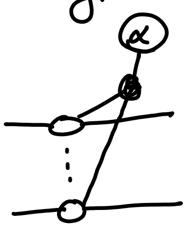
$$:: \ |x\rangle \mapsto \begin{cases} 1 & \text{if } x=0 \\ e^{i\alpha} & \text{if } x=1 \end{cases} = e^{i\alpha\cdot x}$$

k-legged phase gadget:

$\sqrt{2}^{(k-1)}$ 

$$:: \ |x_1..x_k\rangle \mapsto e^{i\alpha\cdot(x_1\oplus..\oplus x_k)}$$

In a diagonal unitary:

$\sqrt{2}^{(k-1)}$ 

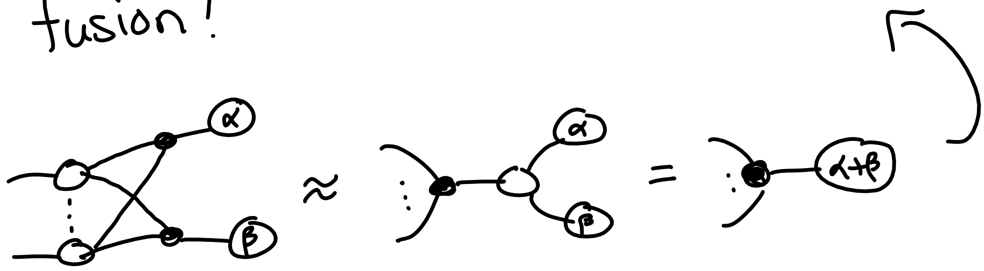
$$:: \ |x_1..x_k\rangle \mapsto e^{i\alpha\cdot(x_1..x_k)}|x_1..x_k\rangle$$

Q: What happens when there is phase folding?

identical connectivity.

fuse using (sc)+(sp).

A: Gadget fusion!



Algorithm: CNOT + phase optimisation.

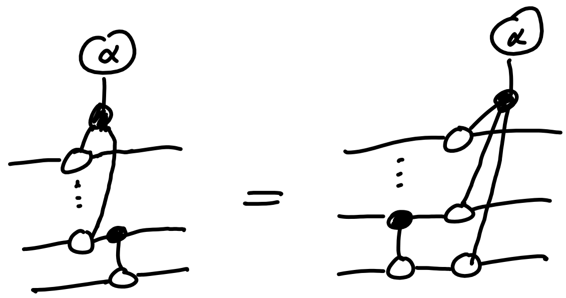
1. unfuse phases and treat as outputs.
2. Compute PNF of phase-free part.
3. perform gadget fusion (* and other phase-poly reductions!)

?? → 4. extract a CNOT + phase circuit.

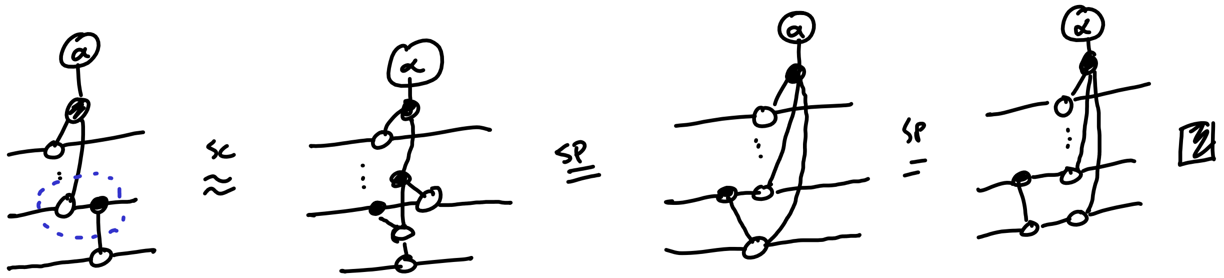
There are choices for step 4.

Naïve approach: "CNOT ladders"

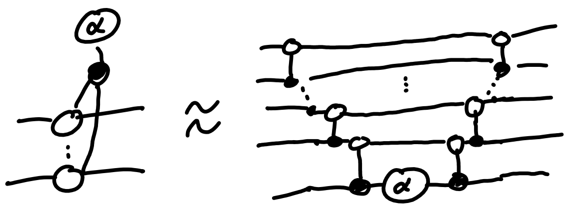
Prop



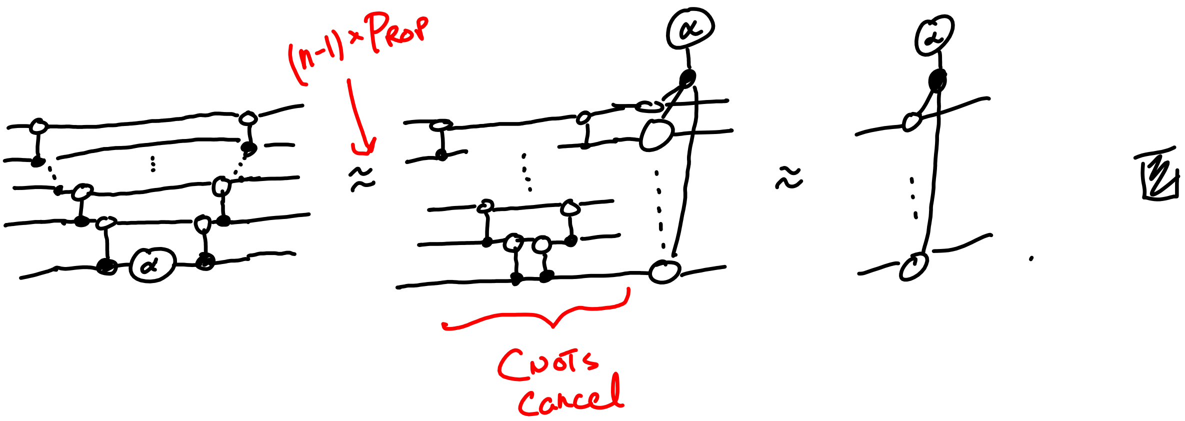
Pf



CoR

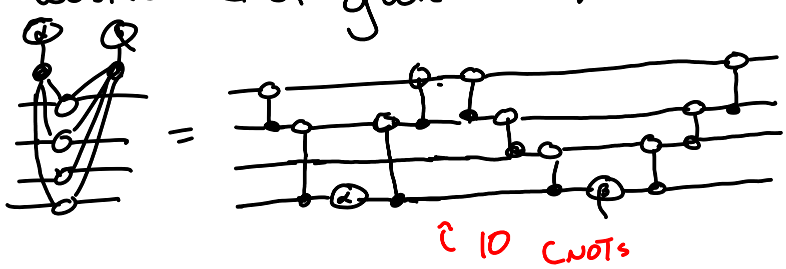


Pf

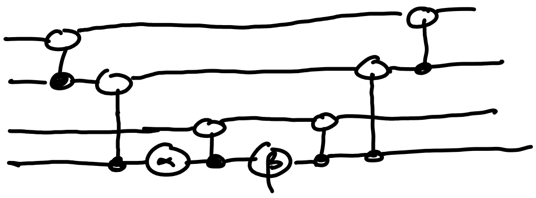


Naïve extraction: 1. unfuse a phase gadget & replace using CoR 1.
2. repeat until no phase gadgets
3. synthesise CNOT circuit from phase-free diag.

* Lots of wasted CNOT gates! e.g.



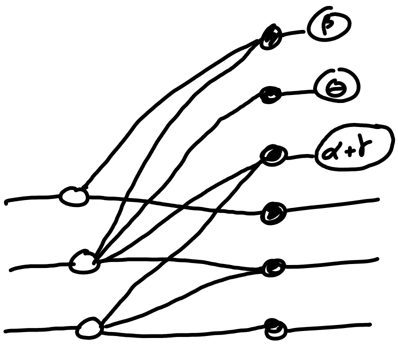
Vs.



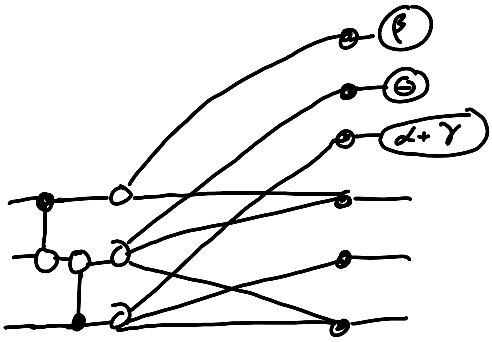
6 CNOTs

"T-par" style extraction [Amy, Maslov, Mosca 2013]

1. write an "extended biadjacency matrix"
2. identify a set of k linearly independent rows
3. reduce each row to a unit vector with column ops.
4. "extract phases" and repeat.



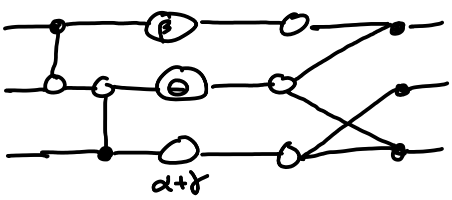
≈ ... ≈



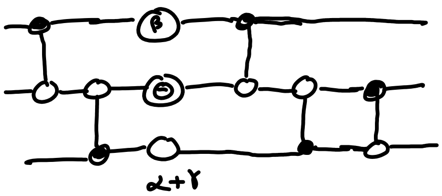
$\overset{sp}{=}$ *

$$
\begin{array}{r}
\text{gadgets} \\ \\ \\ \text{outputs} \\ \\ \\
\end{array}
\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ \hdashline 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}
\xrightarrow{c_2 = c_2 + c_1}
\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}
\xrightarrow{c_2 = c_2 + c_3}
\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \hdashline 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}
$$

* =



≈



$$
\begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}
\xrightarrow{c_2 = c_2 + c_1}
\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}
\xrightarrow{c_2 = c_2 + c_3}
\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}
\xrightarrow{c_3 = c_3 + c_2}
\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}
$$

Pro's: · very good at low non-Clifford depth (i.e. layers of non-Cliff gates).
· gets better with ancillae!

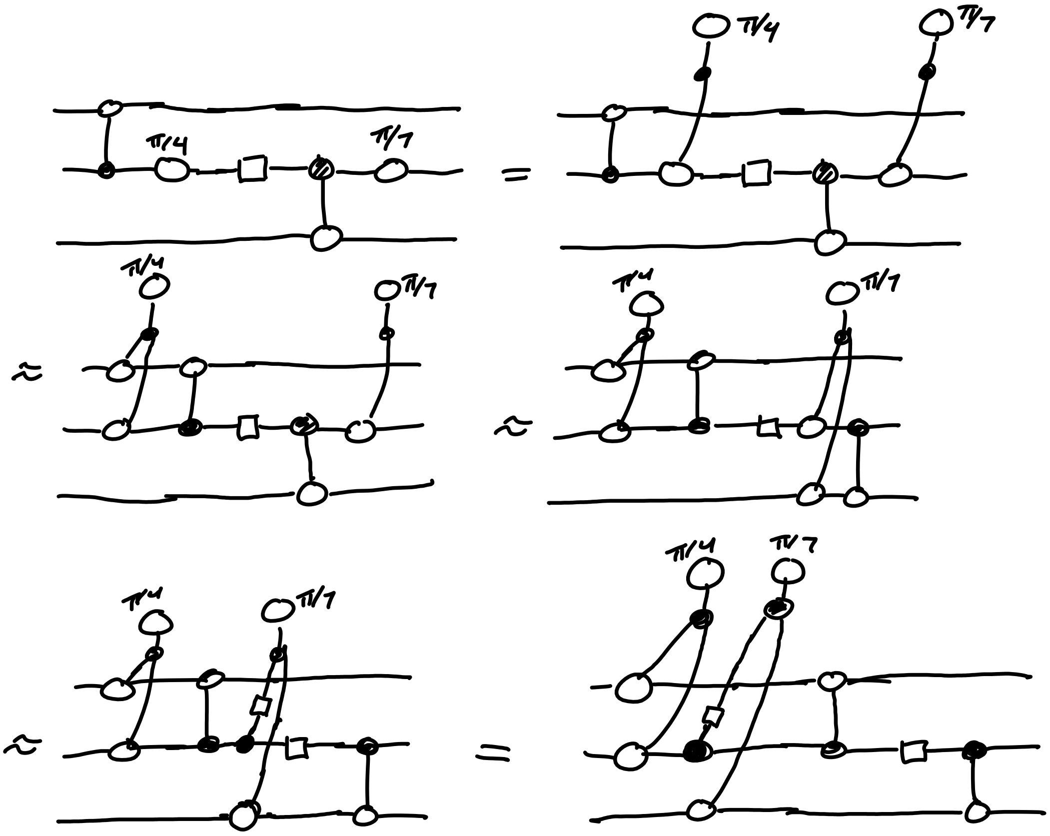
Cons · CNOT count/depth is inconsistent.

* Better for CNOT count: Gray-synth [Amy, Azimzadeh, Mosca 2017]

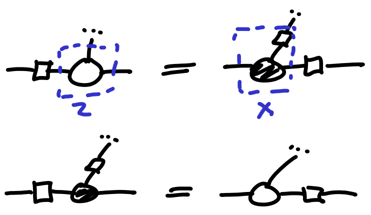
Pauli Gadgets

Clifford + Phase is a universal family.

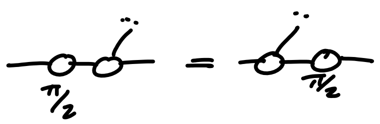
Q: Can we move all the non-Clifford phases out?

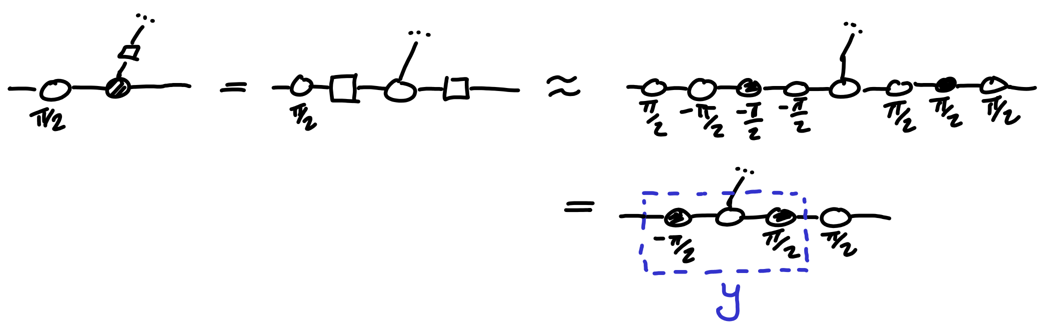


H gates:

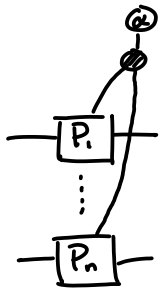


S gates:

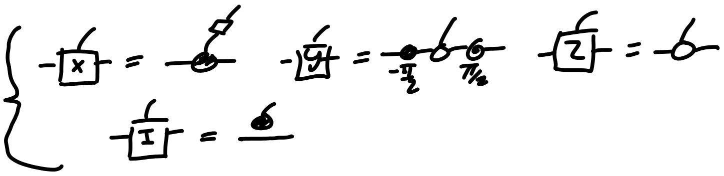




Prop For $\vec{P} = P_1 \otimes \ldots \otimes P_n$ with $P_i \in \{I, X, Y, Z\}$ the map:



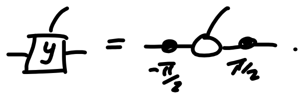
where:



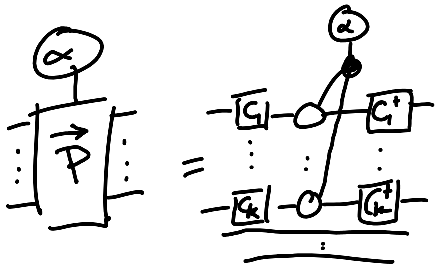
is unitary. It is called the Pauli gadget $\vec{P}(\alpha)$.

Pf Note

and

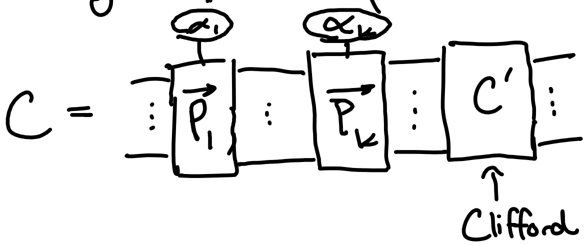


. So



for Cliff. unitaries C_i. Since phase gadgets are unitary, so is $\vec{P}(\alpha)$. ☒

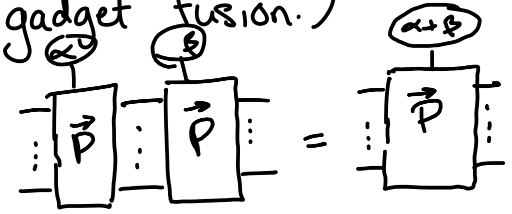
Thm Any Clifford+phase circuit can be written as:



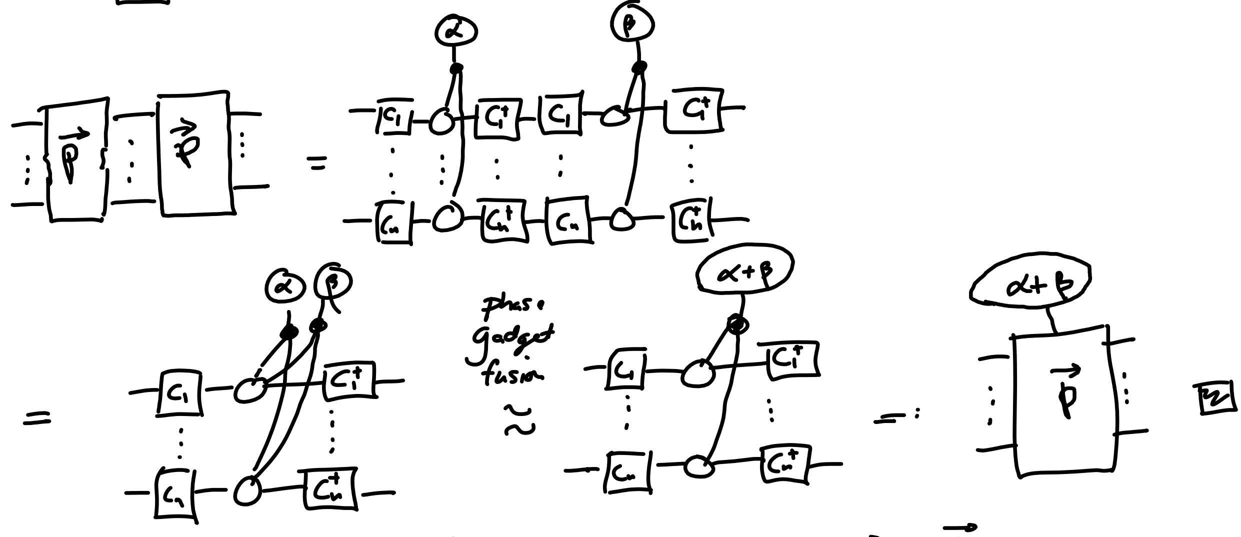
Pf (Idea) • Show Pauli gadgets commute past all Clifford gates.

• Move phases out of C, one at a time. ☒

Prop (Pauli gadget fusion.)



Pf



Prop For Paulis $\vec{P}, \vec{Q}$ if $\vec{P}\vec{Q} = \vec{Q}\vec{P}$, then $\vec{P}(\alpha)\vec{Q}(\beta) = \vec{Q}(\beta)\vec{P}(\alpha)$.

Pf Exercise/book (Hint: it's complementarity!)

Algorithm Pauli "phase folding".

1. Compute Pauli gadget form of a circuit.
2. Commute PG's and combine phases where possible.
3. Merge PG's with Clifford phases into the Clifford part.
4. Repeat until no more reductions.
5. Extract circuit.*

* like with CNOT+phase, there are many options.

Q: What are Pauli gadgets?
A: Exponentials of Pauli matrices.

In general, matrix exponentials are defined as:

$$e^A := \sum_{k=0}^{\infty} \frac{A^k}{k!} \quad \leftarrow \text{Taylor series}$$

Special case: numbers $\quad e^z = \sum_{k=0}^{\infty} \frac{z^k}{k!}$

diagonal matrices: $\quad e^{\mathrm{diag}(a_1,\dots,a_n)} = \sum_{k=0}^{\infty} \frac{\mathrm{diag}(a_1,\dots a_n)^k}{k!}$

$$= \sum_{k=0}^{\infty} \frac{\mathrm{diag}(a_1^k,\dots,a_n^k)}{k!} = \mathrm{diag}(e^{a_1},\dots,e^{a_n})$$

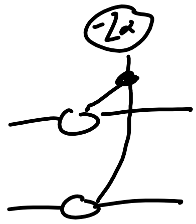
Conjugation: $\quad e^{UAU^\dagger} = \sum_k \frac{(UAU^\dagger)^k}{k!} = \sum_k U \frac{A^k}{k!} U^\dagger = U \sum_k \frac{A^k}{k!} U^\dagger$

$$= U e^A U^\dagger.$$

Ex $\quad Z\otimes Z = \begin{pmatrix}1&0\\0&-1\end{pmatrix}\otimes\begin{pmatrix}1&0\\0&-1\end{pmatrix} = \begin{pmatrix}1&&&\\&-1&&\\&&-1&\\&&&1\end{pmatrix}$

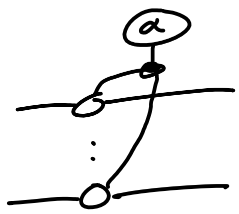
$$U = e^{i\alpha Z\otimes Z} = \begin{pmatrix}e^{i\alpha}&&&\\&e^{-i\alpha}&&\\&&e^{-i\alpha}&\\&&&e^{i\alpha}\end{pmatrix} \propto \begin{pmatrix}1&&&\\&e^{-2i\alpha}&&\\&&e^{-2i\alpha}&\\&&&1\end{pmatrix}$$

$$U|xy\rangle = e^{-2i\alpha\cdot(x\oplus y)}|xy\rangle$$

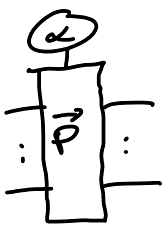


More generally:

Thm $\quad e^{-i\frac{\alpha}{2} Z\otimes\cdots\otimes Z} =$



Cor $\quad e^{-i\frac{\alpha}{2}\vec{P}} =$



Application: Hamiltonian simulation.

Time evolution: $|\Psi_t\rangle = e^{-itH}|\Psi_0\rangle$

Q: Can we design a circuit that implements e^{-itH}?

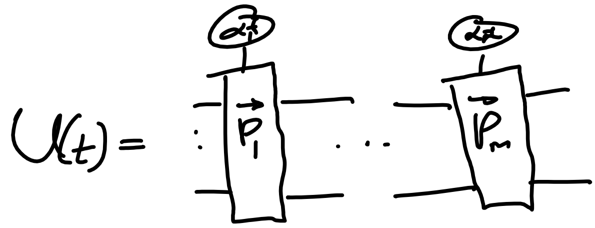
Observation:

Pauli's span the space of all self-adjoint operators on qubits.

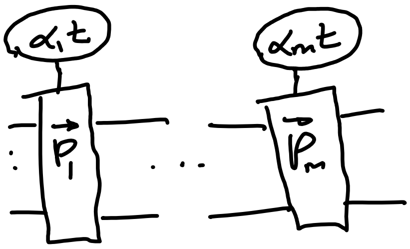
$$H = \frac{1}{2}\sum_{j=1}^{m} \alpha_m \vec{P}_j$$

If all $\vec{P}_j$ commute, then its easy:

$$U = e^{-itH} = e^{-it\frac{\alpha_1}{2}\vec{P}_1} \cdots e^{-it\frac{\alpha_m}{2}\vec{P}_m}.$$



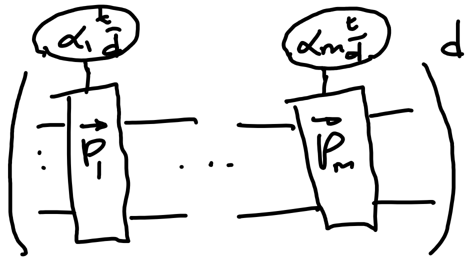
If they do not commute, this is still approximately true.

$U(t) \overset{t^2 K}{\approx}$ 

The trick: make t very small!

$$e^{-itH} = \Big(e^{-t/d\,H}\Big)^{d}$$

approximate this ℓ times

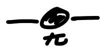
$U(\tfrac{t}{d})^{d} \overset{d\cdot(\frac{t}{d})^2 K}{\approx}$ 

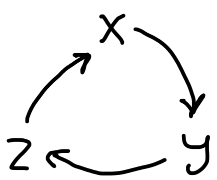
Error: $d\cdot\left(\frac{t}{d}\right)^2\cdot K = \frac{t^2}{d}\cdot K \longrightarrow 0$ as $d \to \infty$.

Lecture 14 | Pauli groups & Stabiliser Theory

The Pauli matrices:

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





$$X^2 = Y^2 = Z^2 = I$$
$$XY = iZ$$

$$\Rightarrow \quad YZ = iX, \; ZX = iY$$
$$\Rightarrow \quad ZY = -iX, \; XZ = -iY, \; YX = -iZ$$
$$\Rightarrow \quad PQ = -QP \quad (P \neq Q, \; P, Q \in \{X, Y, Z\})$$

__Def__ The n-qubit Pauli group

$$\mathcal{P}_n = \{ i^k P_1 \otimes \dots \otimes P_n \mid P_j \in \{I, X, Y, Z\}, \; k \in \{0,1,2,3\} \}$$

We call $\vec{P} \in \mathcal{P}_n$ a <u>Pauli string</u>. It is self-adj. iff $k \in \{0, 2\}$ otherwise $\vec{P}^\dagger = -\vec{P}$.

__Prop__ $\forall \vec{P}, \vec{Q} \in \mathcal{P}_n$, $\vec{P}\vec{Q} = \pm \vec{Q}\vec{P}$.

__Pf__ Follows from the fact that for all $P, Q \in \{I, X, Y, Z\}$, $PQ = \pm QP$. $\boxtimes$

__Ex's__ Count the number of anti-commuting Paulis (even or odd):

$$(X \otimes X \otimes Y)(Z \otimes X \otimes I) = -(Z \otimes X \otimes I)(X \otimes X \otimes Y)$$

$$(X \otimes X \otimes Y)(Z \otimes I \otimes Z) = (Z \otimes I \otimes Z)(X \otimes X \otimes Y)$$

Let $\langle \vec{P}_1, \dots, \vec{P}_m \rangle$ be the subgroup of $\mathcal{P}_n$ generated by products of $\vec{P}_k$.

Def A subgroup $S \subseteq \mathcal{P}_n$ stabilises a state $|\psi\rangle$ if

$\forall \vec{P} \in S.\ \vec{P}|\psi\rangle = |\psi\rangle$ ($|\psi\rangle$ is a $+1$ e-vect of $\vec{P}$)

Let $\mathrm{Stab}(S) = \{ |\psi\rangle \mid \forall \vec{P} \in S.\ \vec{P}|\psi\rangle = |\psi\rangle \}$.

Prop $\mathrm{Stab}(S)$ is a subspace of $(\mathbb{C}^2)^{\otimes n}$, called the stabiliser subspace.

Pf Show $\mathrm{Stab}(S)$ is closed under linear combinations. $\forall \vec{P} \in S$:

$\vec{P}(\alpha|\psi\rangle + \beta|\phi\rangle) = \alpha \vec{P}|\psi\rangle + \beta \vec{P}|\phi\rangle = \alpha|\psi\rangle + \beta|\phi\rangle$. $\boxtimes$

Def A subgroup $S \subseteq \mathcal{P}_n$ where $\mathrm{Stab}(S) \neq \{0\}$ is called a stabiliser subgroup.

Prop If S is a stab. subgroup:

(1) $-I \notin S$

(2) $\forall \vec{P} \in S$, $\vec{P}^\dagger = \vec{P}$

(3) S is commutative

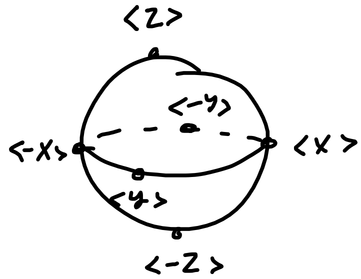
Pf (1) If $-I \in S$, then $(-I)|\psi\rangle = |\psi\rangle \Rightarrow -|\psi\rangle = |\psi\rangle$.
The only possibility is $|\psi\rangle = 0$. Then $(1) \Rightarrow (2)$ and $(2) \Rightarrow (3)$. $\boxtimes$

Ex $\mathcal{P}_1 = \{ i^k P \mid k \in \mathbb{Z},\ P \in \{I, X, Y, Z\}\}$ has 6 stabiliser subgroups: $\langle X \rangle, \langle Y \rangle, \langle Z \rangle, \langle -X \rangle, \langle -Y \rangle, \langle -Z \rangle$

$X|+\rangle = |+\rangle$ $\qquad (-X)|-\rangle = |-\rangle$

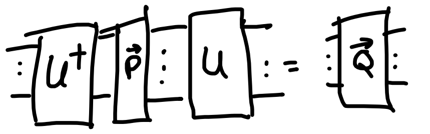
$Y|+i\rangle = |+i\rangle$ $\qquad (-Y)|-i\rangle = |-i\rangle$

$Z|0\rangle = |0\rangle$ $\qquad (-Z)|1\rangle = |1\rangle$



Pushin' PAULIS:

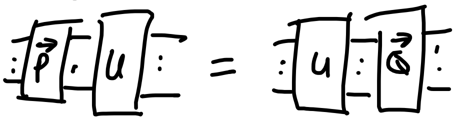
THM For any Clifford unitary U, $\vec{P} \in \mathcal{P}_n$:



$\in \mathcal{P}_n$.

Pf

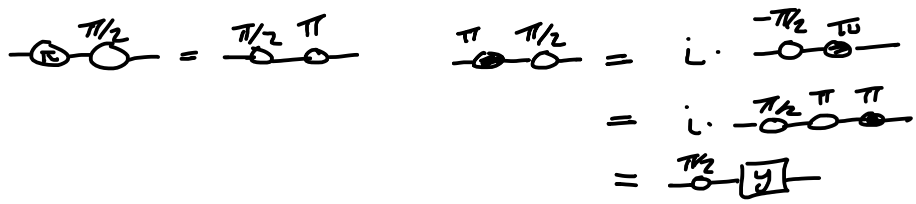
Equivalently, we can "push" Paulis through:



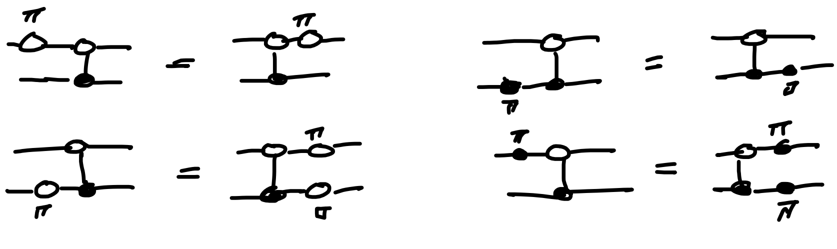
Every $\vec{P}$ is a product of  and , so we can push through Clifford gates 1 at a time:

H:

S:



CNOT:



$\boxed{Z}$

Stabiliser measurements.

Every s.a. $\vec{P}$ defines 2 projectors:

$$\Pi^{(0)}_{\vec{P}} = \tfrac{1}{2}(I + \vec{P}) \qquad \Pi^{(1)}_{\vec{P}} = \tfrac{1}{2}(I - \vec{P})$$

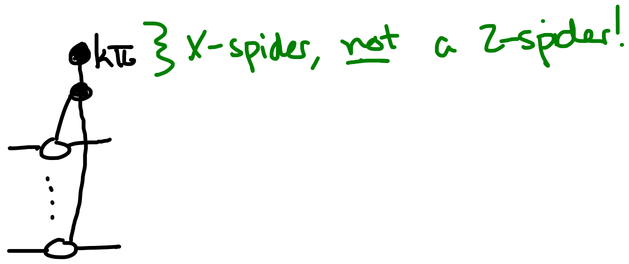
$$\Pi^{(k)}_{\vec{P}} = \tfrac{1}{2}(I + (-1)^k \vec{P})$$

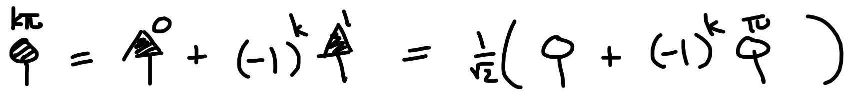
$$\Pi^{(0)}_{\vec{P}} + \Pi^{(1)}_{\vec{P}} = \tfrac{1}{2}(2 \cdot I) = I$$

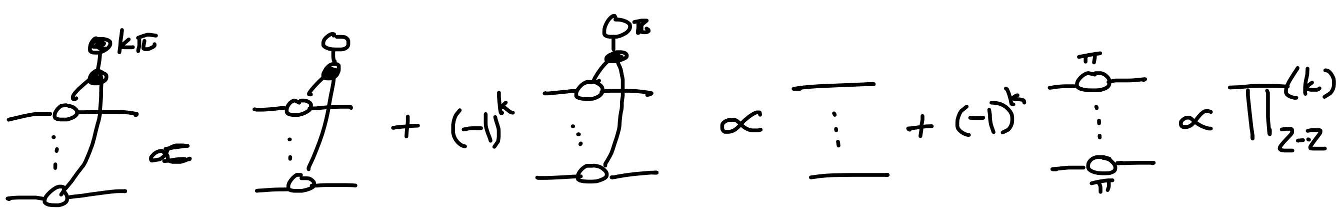
So they define a measurement $M_{\vec{P}} = \{ \Pi^{(k)}_{\vec{P}} \}_{k=0,1}$

If $\vec{P} \in S$, a stabiliser subgp, $M_{\vec{P}}$ is called a <u>stabiliser</u> <u>measurement</u>.

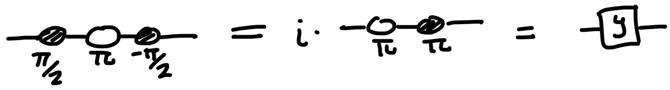
Graphically:

$\Pi^{(k)}_{Z\otimes \cdots \otimes Z} \propto$ 

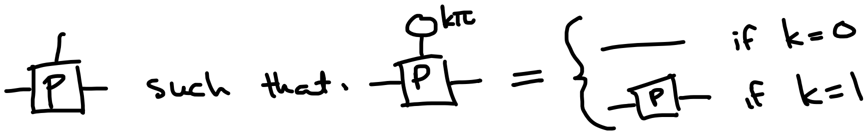




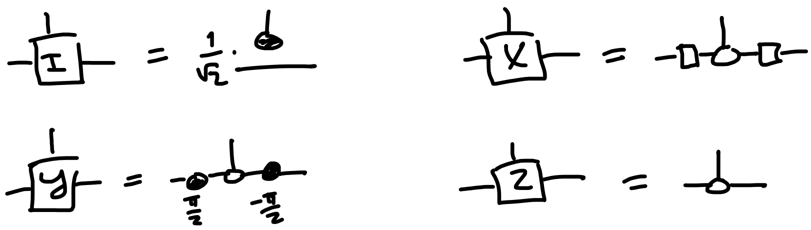
To get other Paulis, we can conjugate by LC's:



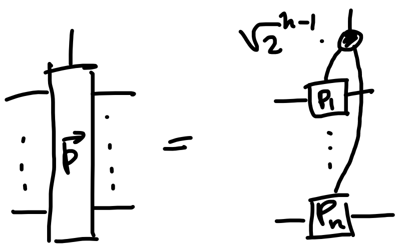
Pauli boxes



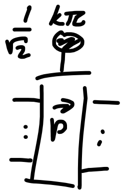
for $k \in \{I, X, Y, Z\}$.



Prop For $\vec{P} = P_1 \otimes \dots \otimes P_n$, let:

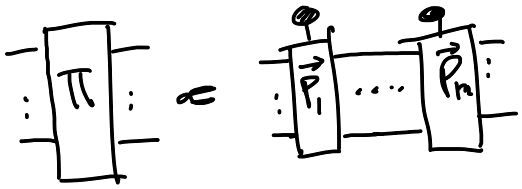


then



$= \Pi^{(k)}_{\vec{P}}$.

Lem For $S = \langle \vec{P}_1, \dots, \vec{P}_m \rangle$:



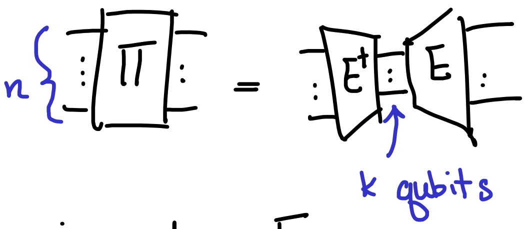
is a projector, and $\operatorname{im} \Pi = \operatorname{Stab}(S)$

Fundamental theorem of stabiliser theory

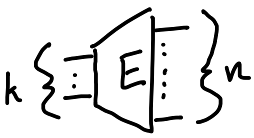
Thm (FTST) For $S = \langle \vec{P}_1, \dots, \vec{P}_m \rangle \subseteq \mathcal{P}_n$ a stabiliser subgp. with m indep. generators:

$$\dim(\operatorname{Stab}(S)) = 2^k \quad \text{for } k = n - m.$$

Pf (idea) Show we can split the projector Π:



for an isometry E. $\boxtimes$



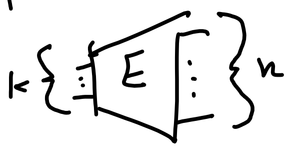
is called an <u>encoder</u> for $\operatorname{Stab}(S)$.

We will use it to treat $\operatorname{Stab}(S)$ as a <u>quantum error correcting code</u>.

Def A <u>stabiliser</u> <u>code</u> is a subspace $\mathrm{Stab}(S) \leq (\mathbb{C}^2)^{\otimes n}$.
for $S = \langle \underbrace{\vec{P}_1, .., \vec{P}_m}_{\text{indep.}} \rangle$ a stabiser group.

It is characterised by 3 parameters:

Physical qubits: n
Logical qubits: k
<u>Code</u> <u>distance</u>: d



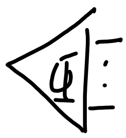
$$[\![n, k, d]\!]$$

Def The <u>weight</u> $|\vec{Q}|$ of $\vec{Q} \in \mathcal{P}_n$ is the # of $Q_j \neq I$.

eg $|X \otimes Y \otimes I| = 2$.

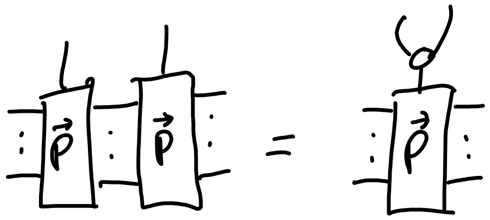
The code distance of $\mathrm{Stab}(S)$ is the smallest weight > 0 of a Pauli str $\vec{Q} \notin S$ where $\vec{P}\vec{Q} = \vec{Q}\vec{P} \; \forall \vec{P} \in S$.

Stabiliser measurements:

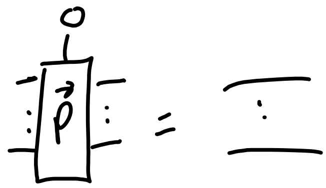
 $\in Stab(S)$

Then perform $M_{\vec{P}_j} = \{$  $\}_{s_j \in 0,1}$

<u>Lem</u>

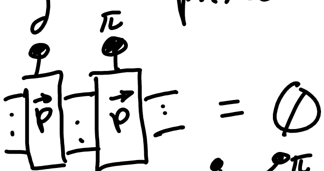


and

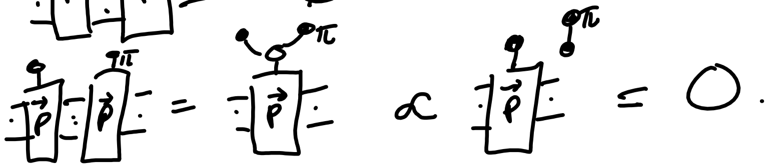


<u>Pf</u> (strong complementarity!)

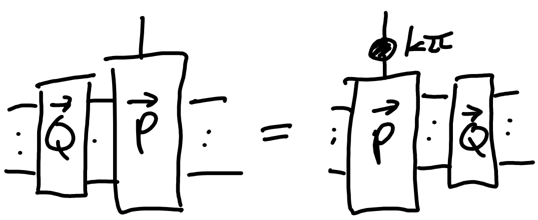
<u>Cor</u>

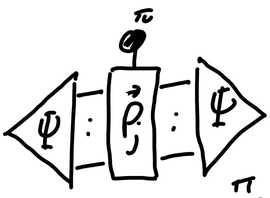


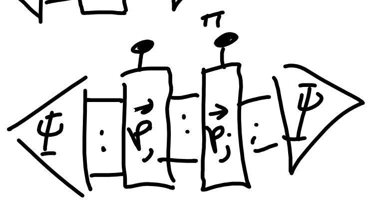
<u>Pf</u>



<u>Lem</u> If $\vec{P}\vec{Q} = (-1)^k \vec{Q}\vec{P}$ then:

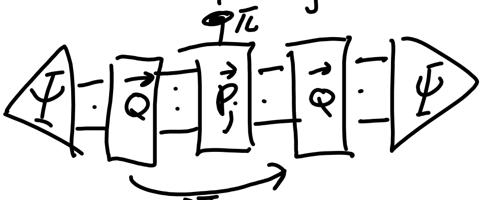


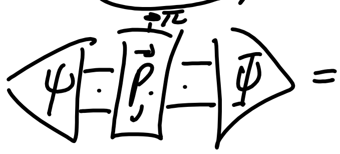
$\mathrm{Prob}(s_j = 1 \,|\, |\Psi\rangle) =$ 

$=$  $= \emptyset$.

$\Rightarrow \mathrm{Prob}(s_j = 0 \,|\, |\Psi\rangle) = 1$.

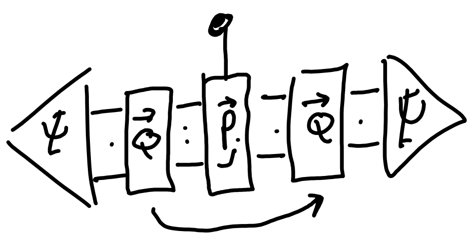
Similarly, if $\vec{Q}$ commutes w/ $\vec{P}_j$:

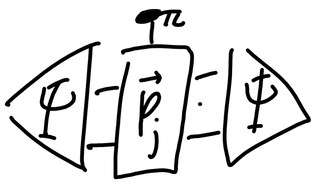
$\mathrm{Prob}(s_j = 1 \,|\, \vec{Q}|\Psi\rangle) =$ 

$=$  $= \emptyset$.

$\Rightarrow \mathrm{Prob}(s_j = 0) = 1$

But if it anti-commutes:

$\mathrm{Prob}(s_j = 0) =$ 

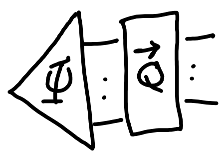
$=$  $= \emptyset$!

$\Rightarrow \mathrm{Prob}(s_j = 1) = 1$

$\Rightarrow$ The outcome s_j detects error $\vec{Q}$. It is called an error syndrome.

An $[\![n,k,d]\!]$ code can detect errors $|\vec{Q}| < d$.

It can also correct errors $|\vec{Q}| < \frac{d}{2}$.

1. Measure stabs:  $\rightsquigarrow$ error syn. $(s_1, .., s_m)$

2. Guess some $\vec{Q}'$ such that $\vec{Q}'\vec{P}_j = (-1)^{s_j}\vec{P}_j\vec{Q}'_j$. ← "decoding" the error
and $|Q'| < \frac{d}{2}$.

Now: $\vec{Q}'\vec{Q}\vec{P}_j = (-1)^{2s_j}\vec{P}_j\vec{Q}'\vec{Q} = \vec{P}_j\vec{Q}'\vec{Q}$.

$|\vec{Q}\vec{Q}'| \leq |\vec{Q}| + |\vec{Q}'| < d$

$\Rightarrow \vec{Q}\vec{Q}' \in S$

3. Apply correction: