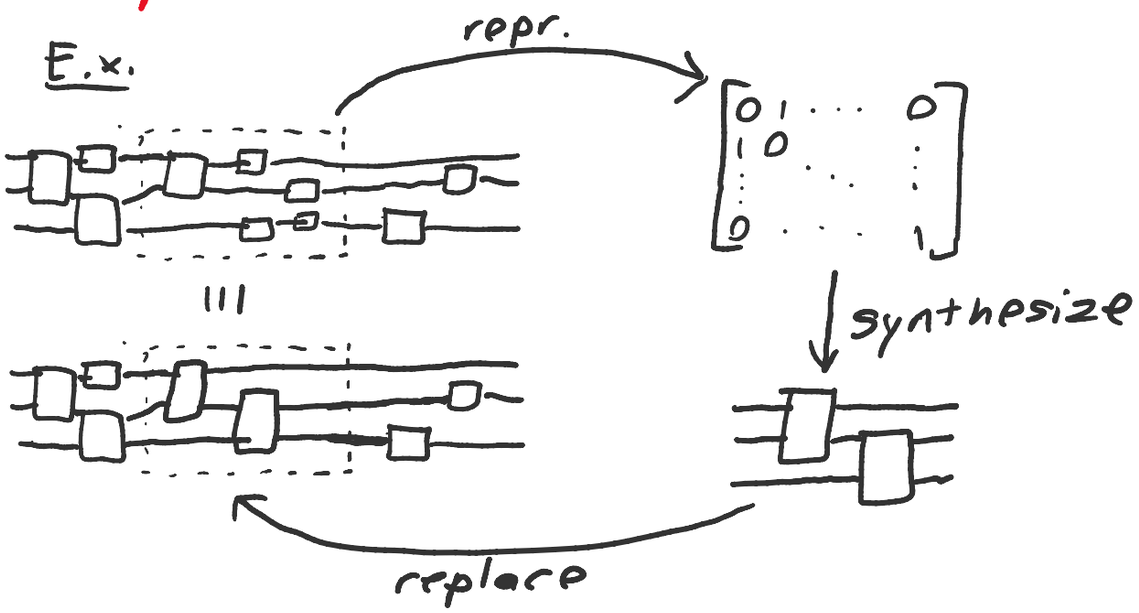
Re-synthesis

Re-synthesis based optimizations take part of a (or a whole) circuit, and synthesize a new one from some representation of it.



In general, don't want to use matrix representations:

1. Exponential in # qubits,
2. Synthesis of arbitrary matrices is highly non-optimal

Effective re-synthesis methods usually involve some efficiently computable representation of a class of circuits, together with optimal or heuristic synthesis for that representation.

Ex.

The CNot gate implements a linear function on $\mathbb{Z}_2^2$.

CNOT $|0\rangle|0\rangle = |0\rangle|0\rangle$
CNOT $|0\rangle|1\rangle = |0\rangle|1\rangle$
CNOT $|1\rangle|0\rangle = |1\rangle|1\rangle$
CNOT $|1\rangle|1\rangle = |1\rangle|0\rangle$

matrix on $\mathbb{Z}_2^2$ →

$\begin{bmatrix}1&0\\1&1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix}$ $\begin{bmatrix}1&0\\1&1\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix}$

$\begin{bmatrix}1&0\\1&1\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}1\\1\end{bmatrix}$ $\begin{bmatrix}1&0\\1&1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix}1\\0\end{bmatrix}$

Representation of CNot circuits

Recall that the general linear group $GL(n,F)$ consists of the $n \times n$ invertible matrices over F

Prop.

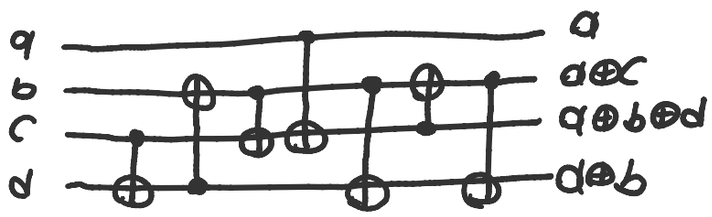
Let C be a circuit over $\{CNOT\}$. Then

$$[\![C]\!]|\vec{x}\rangle = |A\vec{x}\rangle \quad \forall \vec{x} \in \mathbb{Z}_2^n$$

where $A \in GL(n, \mathbb{Z}_2)$

Ex.

Consider the CNOT circuit



The representation as a 4×4 invertible matrix A is

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

Note that

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} a \\ a \oplus c \\ a \oplus b \oplus d \\ a \oplus b \end{bmatrix}$$

(CNOT row operations)

Over $\mathbb{Z}_2$, the only non-trivial row operations are

$R_j \to R_i + R_j$ & $R_i \leftrightarrow R_j$ (add & swap)

These correspond to

$$CNOT_{ij} = E_{ij} \quad \& \quad SWAP_{ij} = CNOT_{ij}\, CNOT_{ji}\, CNOT_{ij} = T_{ij}$$

Synthesis of CNOT circuits

Prop.

Any matrix $A \in GL(n, \mathbb{Z}_2)$ can be synthesized efficiently as a circuit of at most $O(n^2)$ CNOT gates[1]

Pf.

Use Gaussian elimination to compute a sequence of $O(n^2)$ row ops $R_k \cdots R_1 A = I$. Then $A = R_1 \cdots R_k$, where each row operation is 1 or 3 CNOT gates.

Ex.

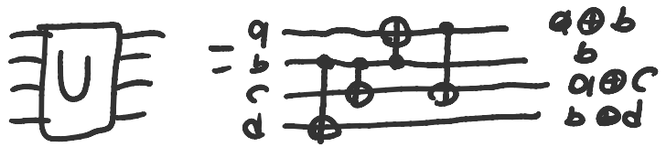
Synthesize $U|\vec{x}\rangle = |A\vec{x}\rangle$ where $A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \xrightarrow{R_3 \to R_1 + R_3} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \xrightarrow{R_1 \to R_1 + R_2} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

$$\downarrow R_3 \to R_2 + R_3$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \xleftarrow{R_4 \to R_2 + R_4} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

Hence



Note: a more efficient sequence exists:

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \xrightarrow{R_1 \to R_1 + R_4} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \xrightarrow{R_3 \to R_1 + R_3} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \xrightarrow{R_4 \to R_2 + R_4} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\therefore$$ 

Optimal* CNOT-synthesis

(Patel-Markov-Hayes, arXiv:quant-ph/0302002)

Patel, Markov & Hayes in 2003 devised an algorithm which produces circuits with $O(n^2/\log n)$ CNOT gates. Given that a simple counting argument shows that there exist $A \in GL(n, \mathbb{Z}_2)$ which require $\Omega(n^2/\log n)$ gates, this is **asymptotically optimal**. In practice though, performance leaves much to be desired...

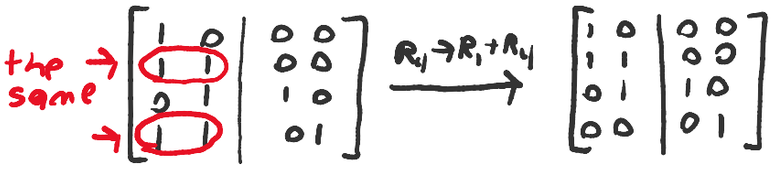
PMH algorithm

1. Divide into groups of k ($\sim \log n/2$) columns
2. For each column group
3. Add any rows which are identical on those colums
4. Perform regular Gaussian elimination on those columns

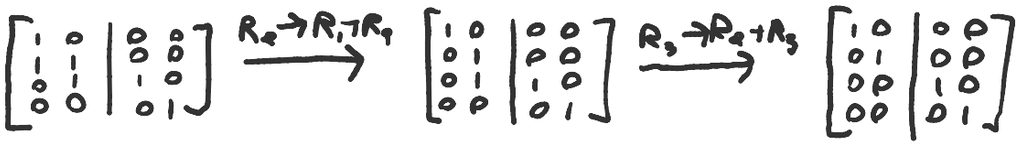
Ex.

We want to synthesize $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$

We first eliminate duplicate rows over the first 2 columns



Now perform Gaussian elimination on first 2 columns



The result uses 3 CNOT gates, vs 4 with regular Gaussian elimination!

Affine permutations

Another class of permutations which can be efficiently represented are affine permutations.

(General Affine group)

The general affine group $GA(n, \mathbb{Z}_2) \simeq GL(n, \mathbb{Z}_2) \ltimes \mathbb{Z}_2^n$ consists of the invertible affine transformations over $\mathbb{Z}_2^n$. That is, for $A \in GL(n, \mathbb{Z}_2)$, $\vec{b} \in \mathbb{Z}_2^n$, then

$$(A, \vec{b}) \in GA(n, \mathbb{Z}_2)$$

$$(A, \vec{b})\vec{x} = A\vec{x} + \vec{b} \qquad \forall \vec{x} \in \mathbb{Z}_2^n$$

Fact

$\{CNOT, X\}$ generates $GA(n, \mathbb{Z}_2)$

To see why, it suffices to note that if

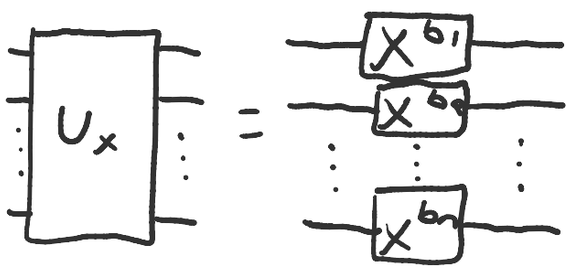
$$U|\vec{x}\rangle = |A\vec{x} + \vec{b}\rangle \,, \quad (A, \vec{b}) \in GA(n, \mathbb{Z}_2)$$

Then we can factorize as $U = U_X U_{CNOT}$ where

$$U_{CNOT}|\vec{x}\rangle = |A\vec{x}\rangle$$

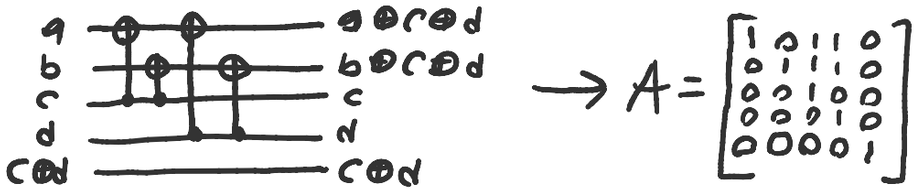
$$U_X|\vec{x}\rangle = |\vec{x} + \vec{b}\rangle$$

Since $A \in GL(n, \mathbb{Z}_2)$, U_{CNOT} can be implemented using only CNOT gates. Moreover,



Other considerations

Often when re-synthesizing, the input state lies in some subspace $V \subseteq \mathbb{Z}_2^n$ due to ancillas. Knowing V gives **extra freedom** in synthesizing A. For example,



$$\rightarrow A = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Synthesizing A gives 4 CNOT gates, but knowing that the 5^{th} qubit starts in the $c \oplus d$ state gives a solution with 2 CNOT gates. In general, we want

$$A \in GL(m, \mathbb{Z}_2) \text{ s.t. } AA_0 = A_1$$

For A_0, A_1 $n \times m$ matrices giving the **input** and **output**, resp. Using a **pseudoinverse** A_0^g we can synthesize

$$A = A_1 A_0^g \in GL(m, \mathbb{Z}_2)$$