量子コンピュータで学ぶ 量子プログラミング入門

MARU LABO

第三章　構成された回路はどのような働きをするのか？

本章の内容　　全体目次 (./Contents.ipynb)

- 第三章　構成された回路はどのような働きをするのか？
  - 一つ一つのゲートはどのような働きをするのか？
  - 量子ゲートとユニタリ行列
  - 基本的な1-qubitゲートと対応するユニタリ行列
  - 演習問題 1
  - ゲートの出力ベクトルを、ユニタリ行列と入力ベクトルの積として計算する
  - 代表的な1-qubitのゲート X, Z, H の働きのまとめ
  - 演習問題 2
- ゲートはどのように組み合わされるのか？
  - 組み合わされたゲートは、どのような働きをするのか？
  - Serialな構成の場合
  - Parallelな構成の場合
  - コントロールの構成の場合
- 数学的準備 2 -- 行列のテンソル積
  - システムのテンソル積
  - CNOTの入力と出力をテンソル積で表示する
  - 演習問題 3
- 構成した回路をシミュレートする
  - 回路の出力ベクトルをシミュレートする
  - 回路全体を一つのユニタリ行列で表す

先に構成した量子回路は、どのような働きをするのだろうか？　それを知る前に、一つ一つのゲートの働きを確認しておこう。

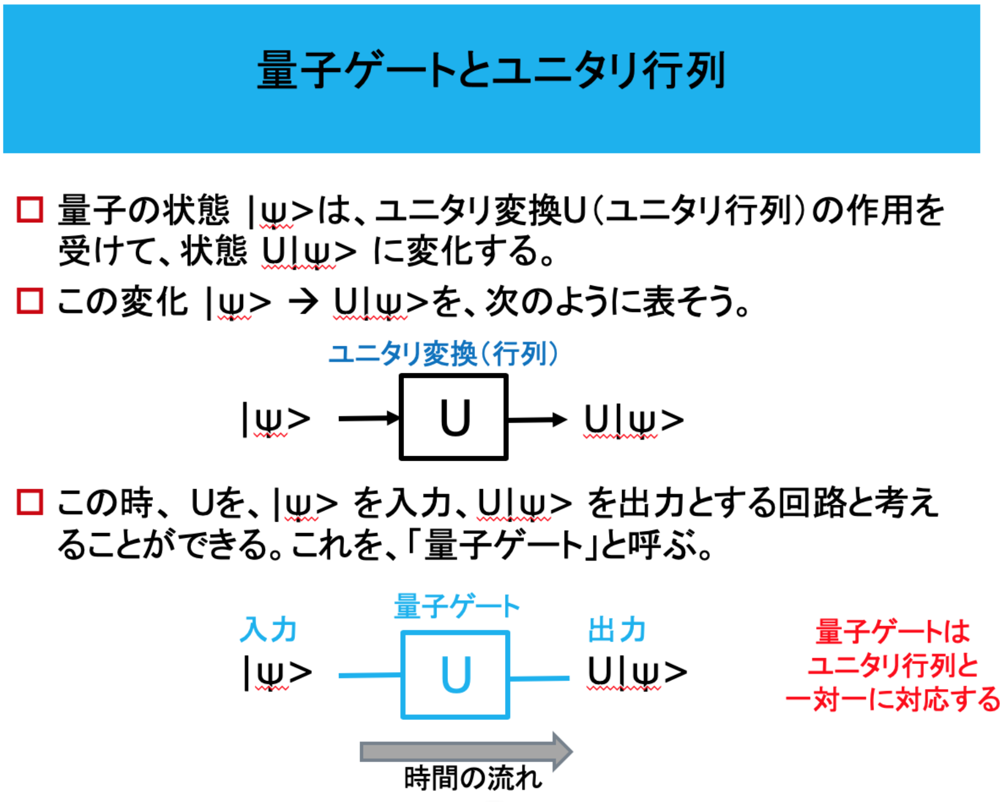
一つ一つのゲートはどのような働きをするのか？

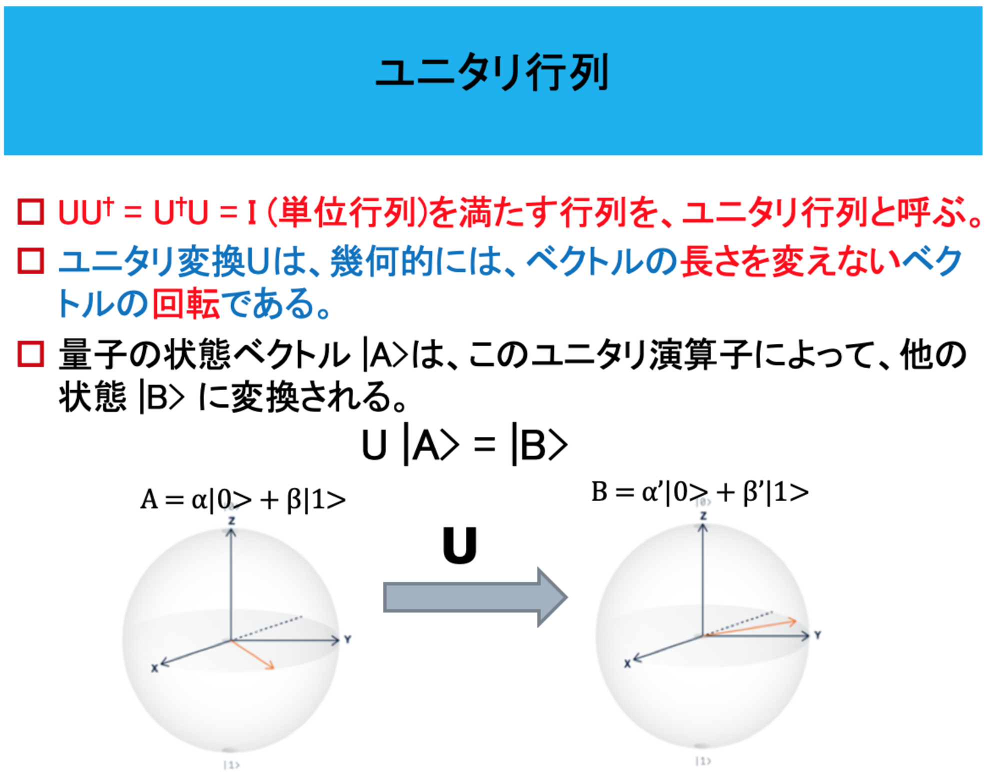
個々の量子ゲートがどのような働きをするのか、その特徴をみておこう。

- **量子ゲートは、入力のqubitたちを出力のqubitたちに変換する。それは、古典的なゲートが、古典ビットの入力を古典ビットの出力に変換するのと同じである。**
- **古典的なゲートは、基本的には、AND, OR, NOTといった「論理的」な演算で定義されるが、量子的なゲートは数学的には、「ユニタリ変換」として定義される。「ユニタリ変換」は、「ユニタリ行列」で定義される。**

- 「ユニタリ行列」は、入力のqubitの「入力ベクトル」を出力のqubitの「出力ベクトル」に変換する。
- 一つの量子ゲートには、一つの「ユニタリ行列」が対応する。
- 古典的なゲートとは異なって、量子ゲートでは入力のqubitの数と出力のqubitの数は等しい。

量子ゲートとユニタリ行列





基本的な1-qubitゲートと対応するユニタリ行列

ゲート名	プログラム上の名前	ゲートの記号	対応するユニタリ行列
Xゲート	'x'	$-\boxed{X}-$	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
Yゲート	'y'	$-\boxed{Y}-$	$\begin{pmatrix} 0 & i \\ -i & 0 \end{pmatrix}$
Zゲート	'z'	$-\boxed{Z}-$	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
Hゲート	'h'	$-\boxed{H}-$	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$
Sゲート	's'	$-\boxed{S}-$	$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$
Tゲート	't'	$-\boxed{T}-$	$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$

演習問題 1

1. 行列Xが$XX^{\dagger} = I$を満たし、ユニタリ行列であることを確かめよ。

2. 行列Zが$ZZ^{\dagger} = I$を満たし、ユニタリ行列であることを確かめよ。

3. 行列Hが$HH^{\dagger} = I$を満たし、ユニタリ行列であることを確かめよ。

ゲートの出力ベクトルを、ユニタリ行列と入力ベクトルの積として計算する

行列Xと入力ベクトル$|0\rangle$の積を計算する

$$X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

行列Zと入力$|0\rangle$の積を計算する

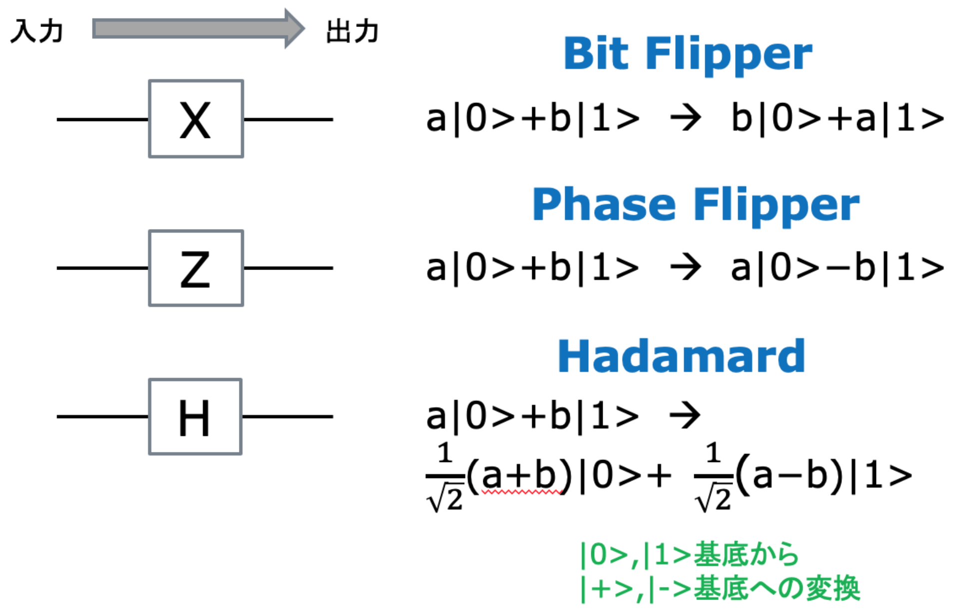
$$Z|0\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

行列Hと入力$|0\rangle$の積を計算する

$$H|0\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

代表的な1-qubitのゲート X, Z, H の働きのまとめ

これらの三つのゲートの働きをしっかり覚えておくと、ずいぶんと回路の働きの計算が楽になる。



演習問題 2

$|0\rangle$は $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $|1\rangle$は $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ で表される。この時、

1. 行列Xと入力ベクトル$|1\rangle$ の積を計算せよ

2. 行列Zと入力ベクトル$|1\rangle$ の積を計算せよ

3. 行列Hと入力ベクトル$|1\rangle$ の積を計算せよ

ゲートはどのように組み合わされるのか？

ゲートから回路を構成する方法は、基本的には、次の三つである。

1. Serialな構成　あるゲートの出力を次のゲートの入力にserialに接続する

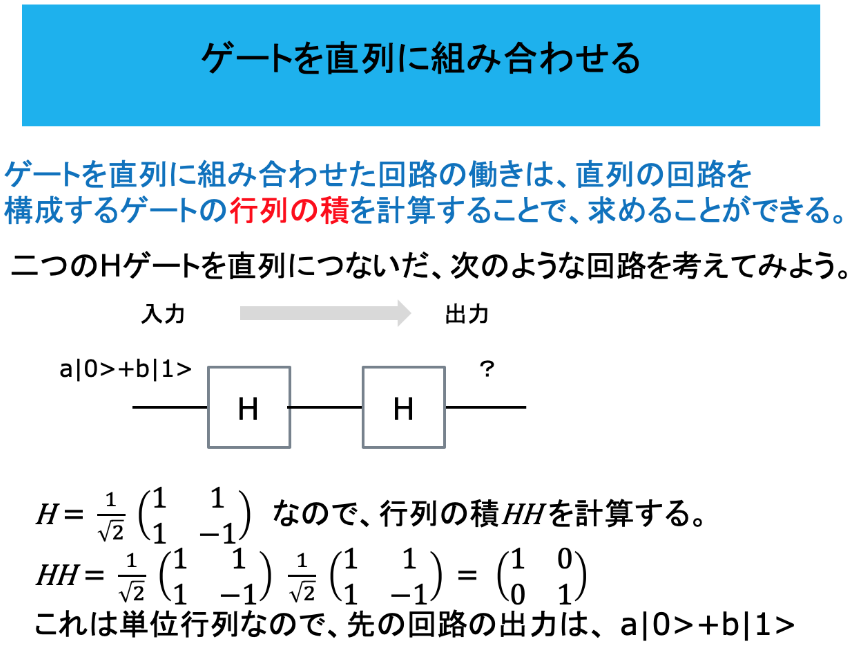
2. Parallelな構成　あるゲートと別のゲートをparallelに構成する

3. コントロールの構成 あるゲートの出力を、parallelに走る別のゲートのコントロールに使用する

組み合わされたゲートは、どのような働きをするのか？

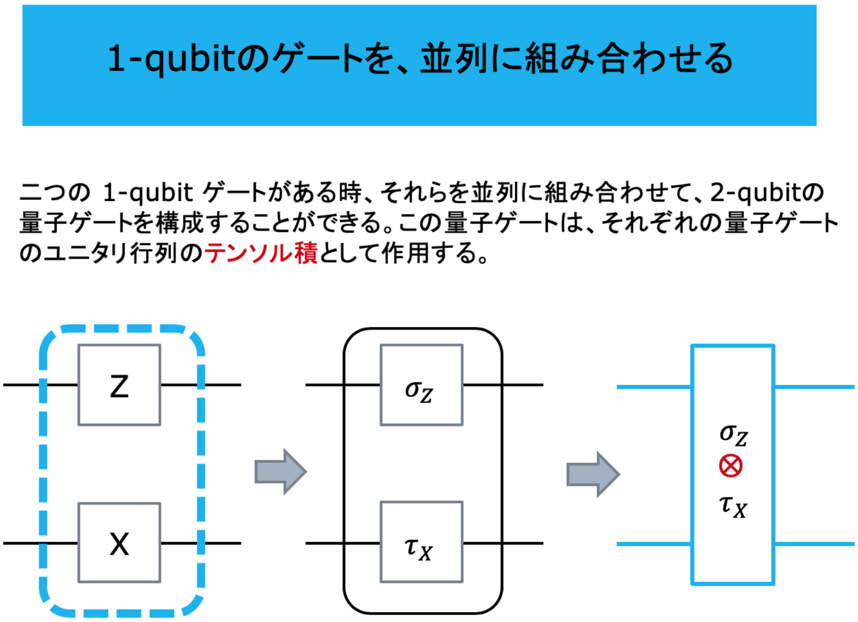
Serialな構成の場合 -- 行列として　積　$U_2 \circ U_1$

ユニタリ行列U_1とU_2でそれぞれ表現される二つのゲートU_1とU_2が、serialに結合された時、この結合されたゲートは、行列U_1とU_2　行列として　積$U_2 \circ U_1$で表現される変換として作用する。



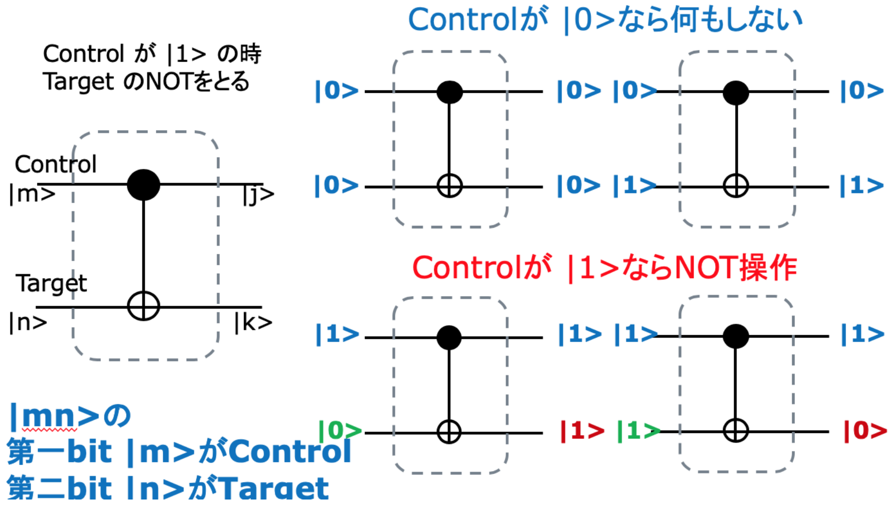
Parallelな構成の場合 -- 行列として　テンソル積　$U_1 \otimes U_2$

ユニタリ行列U_1とU_2でそれぞれ表現される二つのゲートU_1とU_2が、parallelに結合された時、この結合されたゲートは、行列U_1とU_2の行列としてのテンソル積 $U_1 \otimes U_2$で表現される変換として作用する。



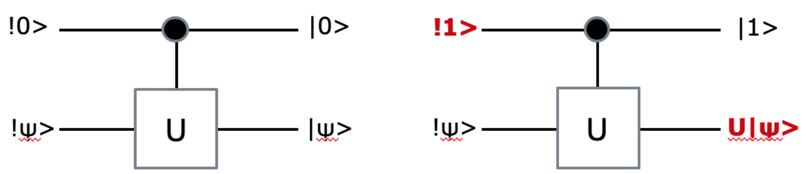
コントロールの構成の場合 -- コントロール $-U$ゲート

2-qubitのゲート
CNOT (Control-NOT)



2-qubitのゲート **Control-U**

コントロール qubit が、|0> の時には、何もせず、(左図)
コントロール qubit が、|1> の時には、ターゲットのqubit
|ψ> にユニタリ演算子 U を適用した U|ψ>を出力する(右図)
ゲートを、**Control-U ゲート**という。



数学的準備 2 -- 行列のテンソル積

行列のテンソル積

行列のテンソル積を次のように定義する。(2x2行列で例示)

$$A \otimes B = \begin{pmatrix} A_{11}B & A_{12}B \\ A_{21}B & A_{22}B \end{pmatrix}$$

Bの成分も書くと、

$$A \otimes B = \begin{pmatrix} \boxed{\begin{matrix} A_{11}B_{11} & A_{11}B_{12} \\ A_{11}B_{21} & A_{11}B_{22} \end{matrix}} & \boxed{\begin{matrix} A_{12}B_{11} & A_{12}B_{12} \\ A_{12}B_{21} & A_{12}B_{22} \end{matrix}} \\ \boxed{\begin{matrix} A_{21}B_{11} & A_{21}B_{12} \\ A_{21}B_{21} & A_{21}B_{22} \end{matrix}} & \boxed{\begin{matrix} A_{22}B_{11} & A_{22}B_{12} \\ A_{22}B_{21} & A_{22}B_{22} \end{matrix}} \end{pmatrix}.$$

行列のテンソル積の例 (1)

A=$\begin{pmatrix} 1 & -1 \\ 0 & 2 \end{pmatrix}$, B=$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ としよう。

$$A\otimes B = \begin{pmatrix} 1 & -1 \\ 0 & 2 \end{pmatrix} \otimes \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 1\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} & -1\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \\ 0\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} & 2\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 2 & -1 & -2 \\ 3 & 4 & -3 & -4 \\ 0 & 0 & 2 & 4 \\ 0 & 0 & 6 & 8 \end{pmatrix}$$

行列のテンソル積の例 (2)

A=$\begin{pmatrix} 1 & -1 \\ 0 & 2 \end{pmatrix}$, B=$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ としよう。

$$B\otimes A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \otimes \begin{pmatrix} 1 & -1 \\ 0 & 2 \end{pmatrix} = \begin{pmatrix} 1\begin{pmatrix} 1 & -1 \\ 0 & 2 \end{pmatrix} & 2\begin{pmatrix} 1 & -1 \\ 0 & 2 \end{pmatrix} \\ 3\begin{pmatrix} 1 & -1 \\ 0 & 2 \end{pmatrix} & 4\begin{pmatrix} 1 & -1 \\ 0 & 2 \end{pmatrix} \end{pmatrix}$$

$$= \begin{pmatrix} 1 & -1 & 2 & -2 \\ 0 & 2 & 0 & 4 \\ 3 & -3 & 4 & -4 \\ 0 & 6 & 0 & 8 \end{pmatrix}$$

テンソル積では、
$A\otimes B \neq B\otimes A$
である

ベクトルのテンソル積

ベクトルは、nx1の行列であるから、

$$\begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \otimes \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} a_1 \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \\ a_2 \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} a_1 b_1 \\ a_1 b_2 \\ a_2 b_1 \\ a_2 b_2 \end{pmatrix}$$

ベクトルのテンソル積の例 (1)

$$\begin{pmatrix} 1 \\ 2 \end{pmatrix} \otimes \begin{pmatrix} 3 \\ 4 \end{pmatrix} = \begin{pmatrix} 1 \begin{pmatrix} 3 \\ 4 \end{pmatrix} \\ 2 \begin{pmatrix} 3 \\ 4 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 3 \\ 4 \\ 6 \\ 8 \end{pmatrix}$$

$$\begin{pmatrix} 3 \\ 4 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 3 \begin{pmatrix} 1 \\ 2 \end{pmatrix} \\ 4 \begin{pmatrix} 1 \\ 2 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 3 \\ 6 \\ 4 \\ 8 \end{pmatrix}$$

ベクトルのテンソル積の例 (2)

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$|01\rangle = |0\rangle \otimes |1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

$$|10\rangle = |1\rangle \otimes |0\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$|11\rangle = |1\rangle \otimes |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

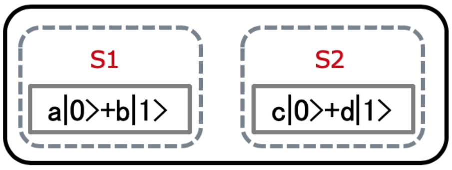
システムのテンソル積

独立した二つのシステムを一つのシステムと考える時、テンソル積で考える。

２つのqubitからなるシステム （テンソル積）

qubit $a|0>+b|1>$ のみを含むシステムをS1、
qubit $c|0>+d|1>$ のみを含むシステムをS2とした時、
このニつを一緒にしたシステムSを テンソル積S1⊗ S2で表す。

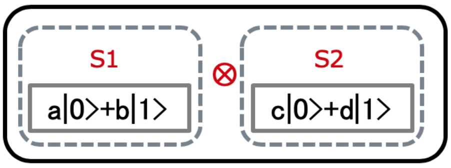
S = S1⊗ S2



この時、次のような計算で、Sの状態を計算する。

$$S = S1 \otimes S2 = (a|0>+b|1>) \otimes (c|0>+d|1>)$$
$$= ac|0>\otimes|0> + ad|0>\otimes|1> + bc|1>\otimes|0> + bd|1>\otimes|1>$$

S = S1⊗ S2

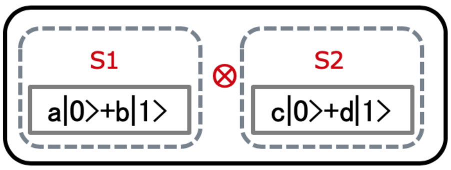


$$= ac|0>\otimes|0> + ad|0>\otimes|1> + bc|1>\otimes|0> + bd|1>\otimes|1>$$

$|x>\otimes|y>$を $|xy>$と表すことにすると、

$$S = S1 \otimes S2 = (a|0>+b|1>) \otimes (c|0>+d|1>)$$
$$= ac|0>\otimes|0> + ad|0>\otimes|1> + bc|1>\otimes|0> + bd|1>\otimes|1>$$
$$= ac|00> + ad|01> + bc|10> + bd|11>$$

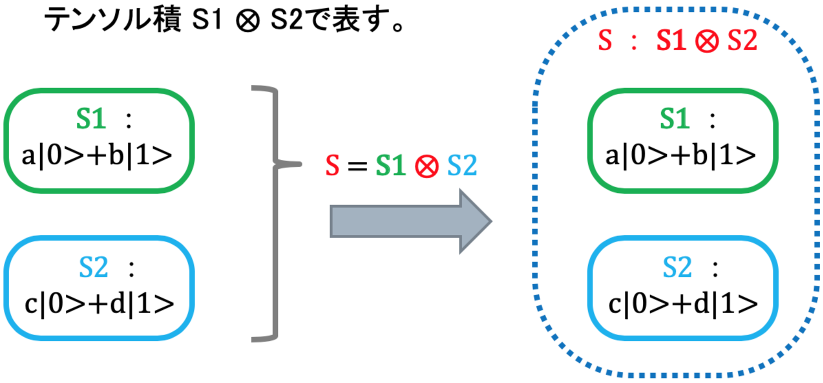
S = S1⊗ S2



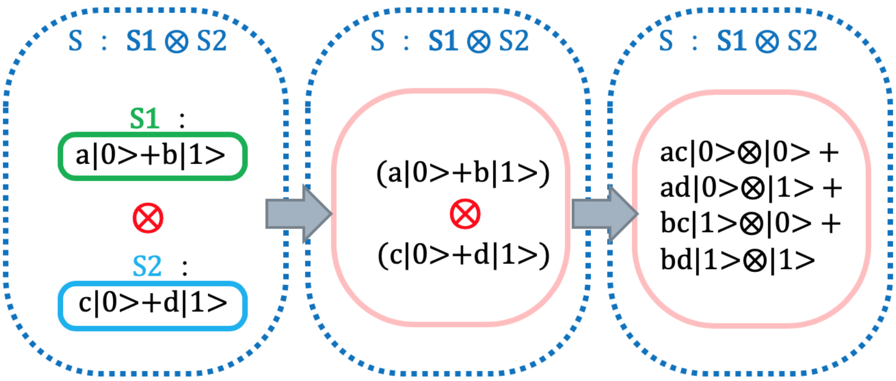
$$= ac|0>\otimes|0> + ad|0>\otimes|1> + bc|1>\otimes|0> + bd|1>\otimes|1>$$
$$= ac|00> + ad|01> + bc|10> + bd|11>$$

もう一度、テンソル積(縦バージョン)

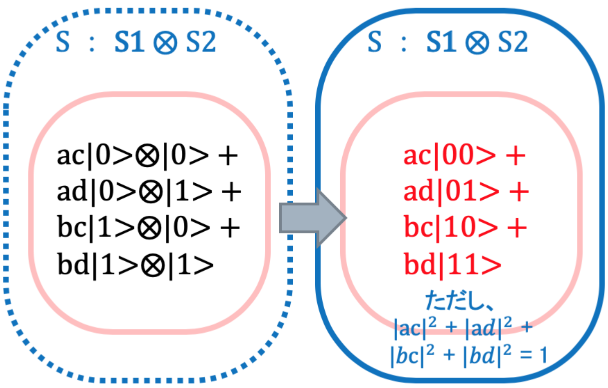
- qubit a|0>+b|1> のみを含むシステムS1、
  qubit c|0>+d|1> のみを含むシステムS2
  があるとしよう。
  この二つを一緒にしたシステムSを
  テンソル積 S1 ⊗ S2で表す。

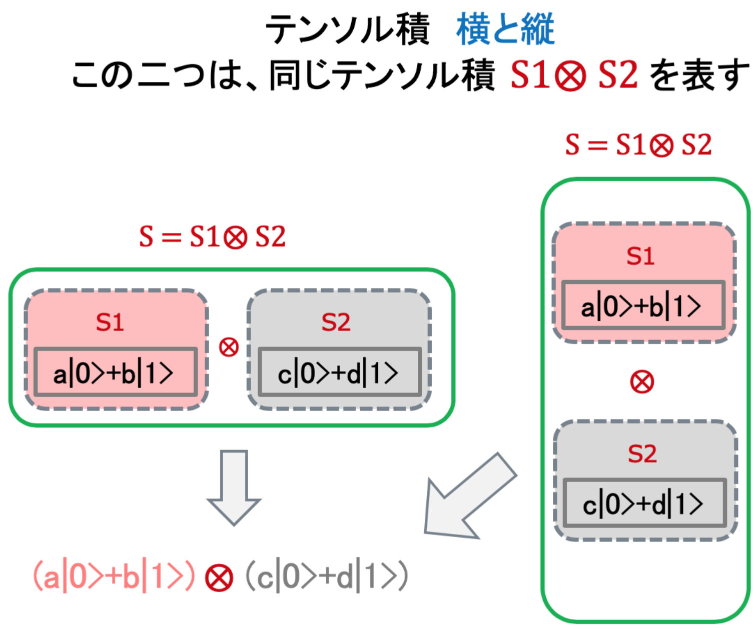


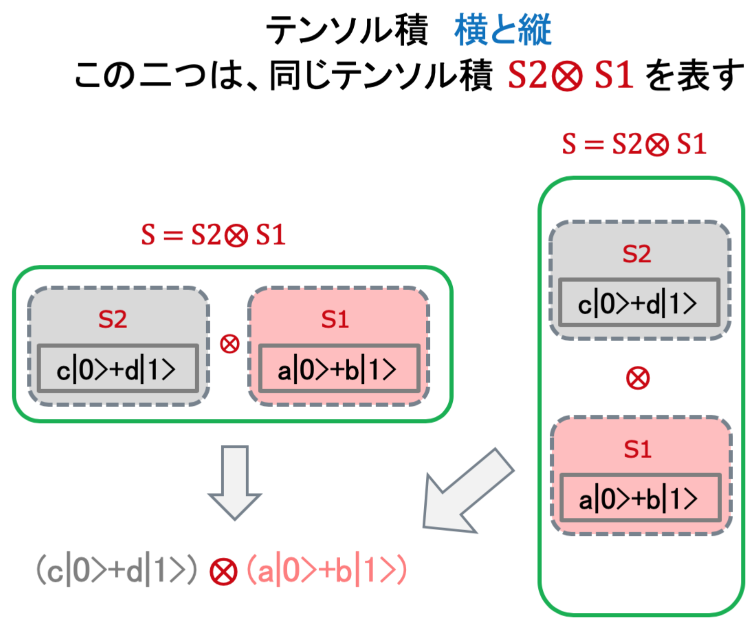
- この時、次のような計算で、Sの状態を計算する。
  S = S1 ⊗ S2 = (a|0>+b|1>) ⊗ (c|0>+d|1>)
  = ac|0>⊗|0> + ad|0>⊗|1> + bc|1>⊗|0> + bd|1>⊗|1>



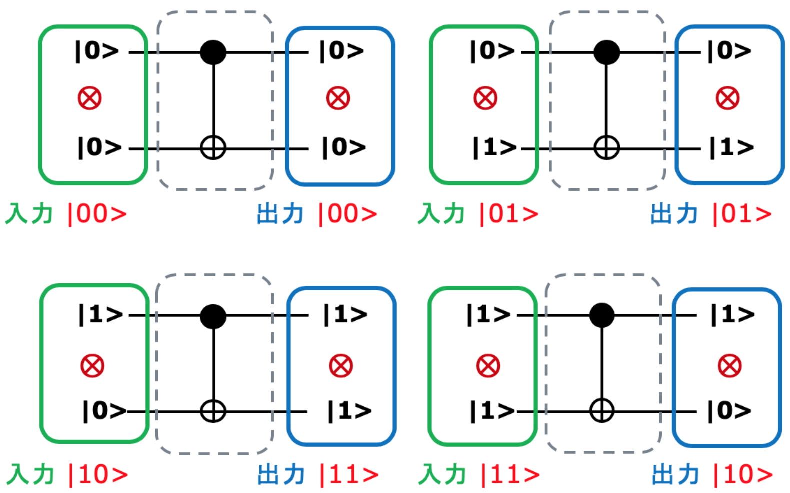
- |a>⊗|b>を |ab>と表すことにする。
  |0>⊗|0> -> |00>, |0>⊗|1> -> |01>,
  |1>⊗|0> -> |10>, |1>⊗|1> -> |11>







CNOTの入力と出力をテンソル積で表示する



|00>→|00>　|01>→|01>　|10>→|11>　|11>→|10>

$$\mathrm{CNOT}\begin{pmatrix}1\\0\\0\\0\end{pmatrix}=\begin{pmatrix}1\\0\\0\\0\end{pmatrix},\ \mathrm{CNOT}\begin{pmatrix}0\\1\\0\\0\end{pmatrix}=\begin{pmatrix}0\\1\\0\\0\end{pmatrix},\ \mathrm{CNOT}\begin{pmatrix}0\\0\\1\\0\end{pmatrix}=\begin{pmatrix}0\\0\\0\\1\end{pmatrix},\ \mathrm{CNOT}\begin{pmatrix}0\\0\\0\\1\end{pmatrix}=\begin{pmatrix}0\\0\\1\\0\end{pmatrix}$$

だから、CNOTを次のように表すことができる。

$$\begin{array}{cc} & \begin{array}{cccc}\text{入力 |00> の出力} & \cdots & & \text{入力 |11> の出力}\end{array} \\ \mathrm{CNOT} = & \begin{pmatrix} \boxed{\begin{matrix}1\\0\\0\\0\end{matrix}} & \boxed{\begin{matrix}0\\1\\0\\0\end{matrix}} & \boxed{\begin{matrix}0\\0\\0\\1\end{matrix}} & \boxed{\begin{matrix}0\\0\\1\\0\end{matrix}} \end{pmatrix}\end{array}$$

演習問題 3

1. 第一のQubitが、$3/5|0\rangle + 4/5|1\rangle$で、第二のQubitが、$1/\sqrt{2}|0\rangle - 1/\sqrt{2}|0\rangle$としよう。この時、二つのqubit が結合した状態を計算せよ。

2. $|0\rangle$は $\begin{pmatrix}1\\0\end{pmatrix}$、$|1\rangle$は $\begin{pmatrix}0\\1\end{pmatrix}$ で表される。この時、

- $|00\rangle = |0\rangle \otimes |0\rangle$を計算せよ
- $|01\rangle = |0\rangle \otimes |1\rangle$を計算せよ
- $|10\rangle = |1\rangle \otimes |0\rangle$を計算せよ
- $|11\rangle = |1\rangle \otimes |1\rangle$を計算せよ
- $|010\rangle = |0\rangle \otimes |1\rangle \otimes |0\rangle$を計算せよ
- $|011\rangle = |0\rangle \otimes |1\rangle \otimes |1\rangle$を計算せよ
- $|100\rangle = |1\rangle \otimes |0\rangle \otimes |0\rangle$を計算せよ
- $|101\rangle = |1\rangle \otimes |0\rangle \otimes |1\rangle$を計算せよ

3. $H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\1 & -1\end{pmatrix}$とする時、$H \otimes H$を求めよ

構成した回路をシミュレートする

Qiskit Aer を使って回路をシミュレートする

Qiskit Aer は量子回路をシミュレートするパッケージである。それは、シミュレーションを実行するための多くの異なったバックエンドを提供する。ここでは、基本的なPythonバージョンを使う

コードのインポート

In [1]:

```
%matplotlib inline
%config InlineBackend.figure_formats = {'png', 'retina'}

import numpy as np
from qiskit import QuantumCircuit, ClassicalRegister, QuantumRegister
from qiskit import execute

# Import Aer
from qiskit import BasicAer
```

回路の出力ベクトルをシミュレートする

量子ゲート H 一つからなる回路の出力ベクトルをシミュレートする

In [2]:

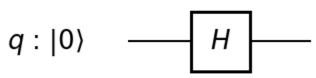
```
# 1 qubitの量子レジスターを生成する
q = QuantumRegister(1, 'q')

# q レジスターに作用する量子回路を生成する
circH = QuantumCircuit(q)

# qubit 0 に H ゲートを追加する。
circH.h(q[0])

# Matplotlib で描画する
circH.draw(output='mpl')
```

Out[2]:



$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ で、$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ である　で、
こ　回路　出力ベクトルは

$$H|0\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

In [4]:

```
# 構成した回路を、状態ベクトル・シミュレータ上でバックエンドで走らせる
backend = BasicAer.get_backend('statevector_simulator')

# 実行用のプログラムを生成する
job = execute(circH, backend)

result = job.result()

outputstate = result.get_statevector(circH, decimals=3)
print(outputstate)
```

```
[0.707+0.j 0.707+0.j]
```

この状態ベクトルは、先の計算で得られた $\frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix}$、すなわち、$\frac{1}{\sqrt{2}}|0\rangle+\frac{1}{\sqrt{2}}|1\rangle$ に他ならない。

量子ゲート X, Y からなる回路の出力ベクトルをシミュレートする

In [5]:

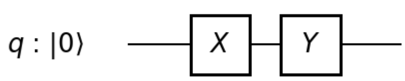
```
# 1 qubitの量子レジスターを生成する
q = QuantumRegister(1, 'q')

# q レジスターに作用する量子回路を生成する
circXY = QuantumCircuit(q)

# qubit 0 に H ゲートを追加する。
circXY.x(q[0])
circXY.y(q[0])

# Matplotlib で描画する
circXY.draw(output='mpl')
```

Out[5]:



$$|0\rangle=\begin{pmatrix}1\\0\end{pmatrix} \text{で、} X=\begin{pmatrix}0&1\\1&0\end{pmatrix}, Y=\begin{pmatrix}0&-i\\i&0\end{pmatrix} \text{である　で}$$

こ　回路　出力ベクトルは

$$YX|0\rangle=\begin{pmatrix}0&-i\\i&0\end{pmatrix}\begin{pmatrix}0&1\\1&0\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}=\begin{pmatrix}-i&0\\0&i\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}=\begin{pmatrix}-i\\0\end{pmatrix}$$

In [6]:

```
# 構成した回路を、状態ベクトル・シミュレータ上でバックエンドで走らせる
backend = BasicAer.get_backend('statevector_simulator')

# 実行用のプログラムを生成する
job = execute(circXY, backend)

result = job.result()

outputstate = result.get_statevector(circXY, decimals=3)
print(outputstate)
```

```
[0.-1.j 0.+0.j]
```

この状態ベクトルは、先の計算で得られた $\begin{pmatrix} -i \\ 0 \end{pmatrix}$、 すなわち、$-i|0\rangle + 0|1\rangle$ に他ならない。

量子ゲート H 二つからなる回路の出力ベクトルをシミュレートする

In [7]:

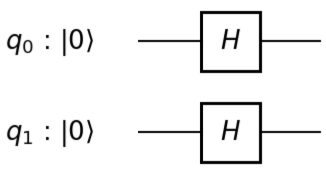
```
# 2 qubitの量子レジスターを生成する
q = QuantumRegister(2, 'q')

# q レジスターに作用する量子回路を生成する
circHH = QuantumCircuit(q)

# qubit 0 に H ゲートを追加する。
circHH.h(q[0])
circHH.h(q[1])

# Matplotlib で描画する
circHH.draw(output='mpl')
```

Out[7]:



これは、二つ 回路が$parallel$に走っている で、テンソル積を使う
入力は、$|0\rangle \otimes |0\rangle = |00\rangle$、回路を表す行列は、$H \otimes H$である

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$ な で

$$H \otimes H|00\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \otimes \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

In [8]:

```
# 構成した回路を、状態ベクトル・シミュレータ上でバックエンドで走らせる
backend = BasicAer.get_backend('statevector_simulator')

# 実行用のプログラムを生成する
job = execute(circHH, backend)

result = job.result()

outputstate = result.get_statevector(circHH, decimals=3)
print(outputstate)
```

```
[0.5+0.j 0.5+0.j 0.5+0.j 0.5+0.j]
```

この状態ベクトルは、先の計算で得られた $\frac{1}{2}\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$、すなわち、$\frac{1}{2}(|0\rangle + |1\rangle + |2\rangle + |3\rangle)$ に他ならない。

CNOT |11> 回路の出力ベクトルをシミュレートする

In [9]:

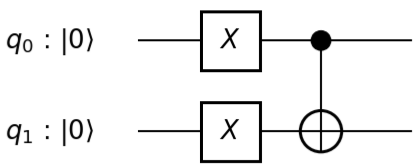
```
# 2 qubitの量子レジスターを生成する
q = QuantumRegister(2, 'q')

# q レジスターに作用する量子回路を生成する
circCNOT11 = QuantumCircuit(q)

# qubit 0 に X ゲートを追加する
circCNOT11.x(q[0])
# qubit 1 に X ゲートを追加する
circCNOT11.x(q[1])
# qubit 1にCNOTゲートを追加する
circCNOT11.cx(q[0], q[1])

# Matplotlib で描画する
circCNOT11.draw(output='mpl')
```

Out[9]:



$$CNOT|11\rangle = |10\rangle$$

$$|11\rangle = |1\rangle \otimes |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

$$|10\rangle = |1\rangle \otimes |0\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$CNOT|11\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

In [10]:

```
# 構成した回路を、状態ベクトル・シミュレータ上でバックエンドで走らせる
backend = BasicAer.get_backend('statevector_simulator')

# 実行用のプログラムを生成する
job = execute(circCNOT11, backend)

result = job.result()

outputstate = result.get_statevector(circCNOT11, decimals=3)
print(outputstate)
```

```
[0.+0.j 1.+0.j 0.+0.j 0.+0.j]
```

この状態ベクトルは、先の計算で得られた $\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$ と等しいか？

実は、この出力は、$0|3\rangle + 1|2\rangle + 0|1\rangle + 0|0\rangle$になっている。

複数のqubitからなるシステムの状態を表現する時に、qiskitで使われているテンソル積の順序は、多くの量子情報理論のテキストとは違っている! ほとんどの物理学のテキストでは（例えば、 Nielsen and Chuangの "Quantum Computation a ndInformation"がそうなのだが）、n個の qubitがあって、それが Q_jのようにj でラベル付けられた時、n-qubit 状態の基底ベクトルは、$Q_0 \otimes Q_1 \otimes \cdots \otimes Q_n$と表現される。**これがqiskitでは違っている！** qiskitでは、n^{th} qubit は、テンソル積の左から数えられる。だから、基底ベクトルは、 $Q_n \otimes \cdots \otimes Q_1 \otimes Q_0$ のようにラベル付けられる。

例えば、qubit 0の状態が 0で, qubit 1が状態 0, qubit 2 が状態 1の時、 qiskit はこの状態を $|100\rangle$と表す。 もちろん、ほとんどのテキストでは、この状態は、$|001\rangle$と表される。

このラベリングの違いは、行列で表現される複数qubitの操作に影響を与える。例えば、qiskitでは、qubit 0がコントロールで、qubit 1がターゲットのcontrolled-X (C_X は、次のように表現される。

$$C_X = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}.$$

回路全体を一つのユニタリ行列で表す

Qiskit Aer also includes a `unitary_simulator` that works *provided all the elements in the circuit are unitary operations*. This backend calculates the $2^n \times 2^n$ matrix representing the gates in the quantum circuit.

量子ゲート H 一つからなる回路をユニタリ行列で表す

In [11]:

```
# Run the quantum circuit on a unitary simulator backend
backend = BasicAer.get_backend('unitary_simulator')
job = execute(circH, backend)
result = job.result()

# Show the results
print(result.get_unitary(circH, decimals=3))
```

```
[[ 0.707+0.j  0.707+0.j]
 [ 0.707+0.j -0.707+0.j]]
```

これは、$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ に等しい

量子ゲート X, Y からなる回路を一つのユニタリ行列で表す

In [12]:

```
# Run the quantum circuit on a unitary simulator backend
backend = BasicAer.get_backend('unitary_simulator')
job = execute(circXY, backend)
result = job.result()

# Show the results
print(result.get_unitary(circXY, decimals=3))
```

```
[[0.-1.j 0.+0.j]
 [0.+0.j 0.+1.j]]
```

これは、$YX = \begin{pmatrix} -i & 0 \\ 0 & i \end{pmatrix}$ に等しい

量子ゲート H 二つからなる回路を一つのユニタリ行列で表す

In [13]:

```
# Run the quantum circuit on a unitary simulator backend
backend = BasicAer.get_backend('unitary_simulator')
job = execute(circHH, backend)
result = job.result()

# Show the results
print(result.get_unitary(circHH, decimals=3))
```

```
[[ 0.5+0.j  0.5+0.j  0.5+0.j  0.5+0.j]
 [ 0.5+0.j -0.5+0.j  0.5+0.j -0.5+0.j]
 [ 0.5+0.j  0.5+0.j -0.5+0.j -0.5+0.j]
 [ 0.5+0.j -0.5+0.j -0.5+0.j  0.5+0.j]]
```

これは、$\frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$に等しい

CNOT 回路を一つのユニタリ行列で表す

In [14]:

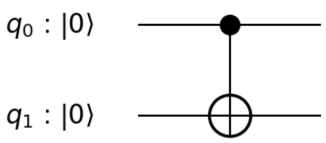
```
# 2 qubitの量子レジスターを生成する
q = QuantumRegister(2, 'q')

# q レジスターに作用する量子回路を生成する
circCNOT = QuantumCircuit(q)

# qubit 1にCNOTゲートを追加する
circCNOT.cx(q[0], q[1])

# Matplotlib で描画する
circCNOT.draw(output='mpl')
```

Out[14]:



In [15]:

```
# Run the quantum circuit on a unitary simulator backend
backend = BasicAer.get_backend('unitary_simulator')
job = execute(circCNOT, backend)
result = job.result()

# Show the results
print(result.get_unitary(circCNOT, decimals=3))
```

```
[[1.+0.j 0.+0.j 0.+0.j 0.+0.j]
 [0.+0.j 0.+0.j 0.+0.j 1.+0.j]
 [0.+0.j 0.+0.j 1.+0.j 0.+0.j]
 [0.+0.j 1.+0.j 0.+0.j 0.+0.j]]
```

これは、$C_X = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$に等しい

前の章へ (./2_circuit.ipynb)　全体目次 (./Contents.ipynb)　次の章

[へ (./4_measurement.ipynb)](./4_measurement.ipynb)

In [ ]:

```

```