Bloque 3: Algoritmos Cuánticos

Alumno: Álvaro Manuel Aparicio Morales

I Certificado de Extensión Universitaria en Computación Cuántica (2024-2025)

Ejercicio 49

Para la resolución de este ejercicio, se ha desarrollado una función para comprobar que dos operadores unitarios son equivalentes mediante el producto de un operador del circuito 1 con el traspuesto conjugado del circuito 2 y la comprobación de que esa multiplicación es igual a la identidad.

```
from qiskit import QuantumCircuit, quantum_info
import numpy as np
```

```
def operators_equivalence(qc_1,qc_2):
    unitary_operator_qc1 = quantum_info.Operator(qc_1)
    unitary_operator_qc2 = quantum_info.Operator(qc_2)

    # Para comprobar si ambos operadores son idénticos el producto del operador del circuito 1
    # con el traspueto conjugado del circuito 2, debe dar la identidad

    unitary_operator_matrix_qc1 = unitary_operator_qc1.data
    unitary_operator_matrix_qc2 = unitary_operator_qc2.data

    print(unitary_operator_matrix_qc1)
    print(unitary_operator_matrix_qc2)

    identity = np.identity(unitary_operator_matrix_qc1.shape[0]) # Obtenemos el tamaño de la matriz para generar la identidad
    operators_products = unitary_operator_matrix_qc1 @ (np.conjugate(unitary_operator_matrix_qc2.T))
    equivalence = np.allclose(identity, operators_products,atol=1e-9)
    return equivalence
```

```
# Circuito 14.3.3.1
qc1_1 = QuantumCircuit(2)
qc1_1.cz(0,1)
qc1_2 = QuantumCircuit(2)
qc1_2.cz(1,0)
equivalence = operators_equivalence(qc_1=qc1_1, qc_2=qc1_2)
print("Result of equivalence is: ", equivalence)
```

```
[[ 1.+0.j  0.+0.j  0.+0.j  0.+0.j]
 [ 0.+0.j  1.+0.j  0.+0.j  0.+0.j]
 [ 0.+0.j  0.+0.j  1.+0.j  0.+0.j]
 [ 0.+0.j  0.+0.j  0.+0.j -1.+0.j]]
[[ 1.+0.j  0.+0.j  0.+0.j  0.+0.j]
 [ 0.+0.j  1.+0.j  0.+0.j  0.+0.j]
 [ 0.+0.j  0.+0.j  1.+0.j  0.+0.j]
 [ 0.+0.j  0.+0.j  0.+0.j -1.+0.j]]
Result of equivalence is:  True
```

```
# Circuito 14.3.3.2
qc2_1 = QuantumCircuit(2)
qc2_1.h(0)
qc2_1.h(1)
qc2_1.cx(0,1) # cnot
qc2_1.h(0)
qc2_1.h(1)

qc2_2 = QuantumCircuit(2)
qc2_2.cx(1,0) # cnot()
equivalence = operators_equivalence(qc_1=qc2_1, qc_2=qc2_2)
print("Result of equivalence is: ", equivalence)
```

```
[[1.+0.j 0.+0.j 0.+0.j 0.+0.j]
 [0.+0.j 1.+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]]
[[1.+0.j 0.+0.j 0.+0.j 0.+0.j]
 [0.+0.j 1.+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]]
Result of equivalence is:  True
```

```
# Circuito 14.3.3.3
qc3_1 = QuantumCircuit(2)
qc3_1.cx(1,0)
qc3_1.cx(0,1)
qc3_1.cx(1,0)
qc3_2 = QuantumCircuit(2)
qc3_2.swap(0,1)
equivalence = operators_equivalence(qc_1=qc3_1, qc_2=qc3_2)
print("Result of equivalence is: ", equivalence)
```

```
[[1.+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]
 [0.+0.j 0.+0.j 0.+0.j 1.+0.j]]
[[1.+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]
 [0.+0.j 0.+0.j 0.+0.j 1.+0.j]]
Result of equivalence is:  True
```

```
# Circuito 14.3.3.4
from qiskit.circuit.library import U1Gate

def get_ckmatrix (phi):
    ckmatrix = np.array([
        [1, 0, 0, 0],
        [0, 1, 0, 0],
        [0, 0, np.exp(1j * phi), 0],
        [0, 0, 0, np.exp(1j * phi)]
    ])
    return ckmatrix

phase = np.pi

qc4_1 = QuantumCircuit(2)
ck_operator = quantum_info.Operator(get_ckmatrix(phase)) # Construimos el operador CKphi para cualquier fase
qc4_1.append(ck_operator,[0,1])

qc4_2 = QuantumCircuit(2)
qc4_2.append(U1Gate(phase),[1])

equivalence = operators_equivalence(qc_1=qc4_1, qc_2=qc4_2)
print("Result of equivalence is: ", equivalence)
```

```
[[ 1.+0.0000000e+00j  0.+0.0000000e+00j  0.+0.0000000e+00j
   0.+0.0000000e+00j]
 [ 0.+0.0000000e+00j  1.+0.0000000e+00j  0.+0.0000000e+00j
   0.+0.0000000e+00j]
 [ 0.+0.0000000e+00j  0.+0.0000000e+00j -1.+1.2246468e-16j
   0.+0.0000000e+00j]
 [ 0.+0.0000000e+00j  0.+0.0000000e+00j  0.+0.0000000e+00j
  -1.+1.2246468e-16j]]
[[ 1.+0.0000000e+00j  0.+0.0000000e+00j  0.+0.0000000e+00j
   0.+0.0000000e+00j]
 [ 0.+0.0000000e+00j  1.+0.0000000e+00j  0.+0.0000000e+00j
   0.+0.0000000e+00j]
 [ 0.+0.0000000e+00j  0.+0.0000000e+00j -1.+1.2246468e-16j
   0.+0.0000000e+00j]
 [ 0.+0.0000000e+00j  0.+0.0000000e+00j  0.+0.0000000e+00j
  -1.+1.2246468e-16j]]
Result of equivalence is:  True
```