Problem Set 2 - Algebra of quantum operations

Computação Quântica 2019/2020

February 14, 2020

For this set, it is recommended that you start with the functions from Exercise 4 of Problem Set 1. A script with the functions can be downloaded here. You might find the documentation page for the `Data.Complex` Haskell package helpful for operations regarding complex numbers.

A single-qubit state $|q\rangle$ can be in a superposition of basis states $|0\rangle$ and $|1\rangle$:

$$|q\rangle = \alpha|0\rangle + \beta|1\rangle \tag{1}$$

where α and β are complex coefficients with normalization $|\alpha|^2 + |\beta|^2 = 1$. The bra-ket notation is used to simplify the description of the quantum state in a complex vector space; $|0\rangle$ and $|1\rangle$ are shorthand for column vectors:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}; \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

In Haskell, matrices may be represented as a list of lists:

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \mapsto \texttt{[[1,0],[0,-1]]},$$

Column vectors may be represented with the same configuration:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \mapsto \texttt{[[1],[0]]}\,; \;\; |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \mapsto \texttt{[[0],[1]]}\,.$$

1. The Hadamard gate, described as a matrix:

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

may be used to map a state from the computational basis state $|0\rangle$ to the superposition basis $|+\rangle$, or from the state $|1\rangle$ to $|-\rangle$. Using the matrix multiplication function from exercise 4 of Problem Set 1:

```
gate :: [[Complex Float]] -> [[Complex Float]] -> [[Complex Float]]
```

   (a) Write the vector representation of the state $|+\rangle = H\,|0\rangle$.

   (b) Write the vector representation of the state $|-\rangle = H\,|1\rangle$.

   (c) Write both states as a superposition of basis states $|0\rangle$ and $|1\rangle$, as presented in equation (1).

2. The joint state of a system of qubits is described by the tensor product $\otimes$. For two (separable) qubit states $|q_0\rangle$ and $|q_1\rangle$, the joint state of the system may be written in the bra-ket notation with an implicit tensor product:

$$|q_0\rangle \otimes |q_1\rangle = |q_0\rangle|q_1\rangle = |q_0\, q_1\rangle$$

   Using the tensor product function from Problem Set 1:

```
tensor :: [[Complex Float]] -> [[Complex Float]] -> [[Complex Float]]
```

   (a) Write the vector representation of states $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$.

   (b) Write the vector representation of the state $|010\rangle$.

3. In a complex vector space $\mathbb{C}^n$, the norm of a vector is expressed as:

$$\|\boldsymbol{v}\| := \sqrt{|v_1|^2 + \cdots + |v_n|^2} = \sqrt{v_1\bar{v}_1 + \cdots + v_n\bar{v}_n}.$$

   (a) Implement a function to determine the norm of a vector $\mathbb{C}^n$:

```
norm :: [[Complex Float]] -> Complex Float
```

   (b) Implement a function:

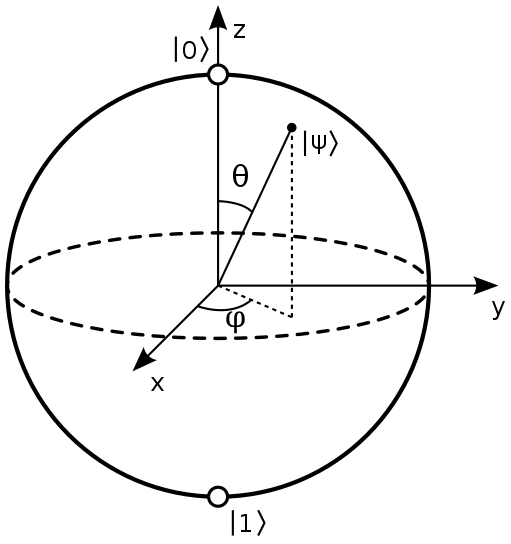
```
normalize :: [[Complex Float]] -> [[Complex Float]]
```

that takes a column vector $\boldsymbol{v}$ representing an n-qubit quantum state, and returns a state $\boldsymbol{v}/\left\|\boldsymbol{v}\right\|$ with normalized coefficients (i.e. such that the sum of the absolute squares over all coefficients is equal to 1).

4. For single-qubit states, the normalisation of probability amplitudes, $|\alpha|^2 + |\beta|^2 = 1$, allows for an alternative description:

$$|\psi\rangle = \cos(\theta/2)|0\rangle + \sin(\theta/2)e^{i\varphi}|1\rangle \tag{2}$$

where $0 \leq \theta \leq \pi$ and $0 \leq \varphi < 2\pi$. From this, it is clear that there is a one-to-one correspondence between single-qubit states ($\mathbb{C}^2$) and the points on the surface of a unit sphere ($\mathbb{R}^3$). This is called the Bloch sphere representation of a qubit state.



Implement a function:

```
bloch :: [[Complex Float]] -> (Float, Float)
```

that takes a column vector describing a single-qubit state, and returns parameters (θ, ϕ) of the Bloch sphere representation.

5. Matrices representing quantum operations are always *unitary*. A general form of a unitary matrix representing single qubit operations may be expressed as:

$$U(\theta, \phi, \lambda) = \begin{pmatrix} \cos(\theta/2) & -e^{i\lambda}\sin(\theta/2) \\ e^{i\phi}\sin(\theta/2) & e^{i\lambda+i\phi}\cos(\theta/2) \end{pmatrix}$$

with $0 \leq \theta \leq \pi\,,\ 0 \leq \phi < 2\pi$ and $0 \leq \lambda < 2\pi$.

(a) Implement a function

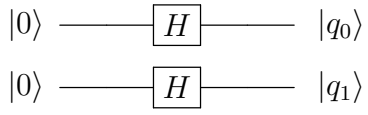
```
u3 :: (Float, Float, Float) -> [[Complex Float]]
```

that takes (θ, ϕ, λ) as arguments and returns matrix U.

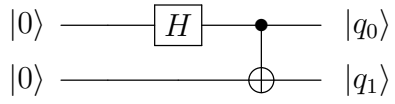
(b) Write the matrix expressions of the Pauli-Z and Hadamard gates as instances of `u3`.

6. In the quantum circuit model of quantum computation, qubits are represented as horizontal lines, with a sequence of boxes over n-lines representing a sequence of n-qubit quantum operations being performed in a *left-to-right* order. Using the functions `gate` and `tensor`:

(a) Write the state of the two-qubit system $|q_0\, q_1\rangle$ after implementation of two parallel Hadamard gates:



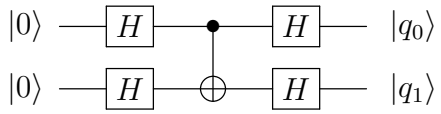
(b) Write the state of the two-qubit system $|q_0\, q_1\rangle$:



Note that the $CNOT$ gate is expressed as the matrix:

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

(c) Build a truth table (i.e. result over input states $|00\rangle, |01\rangle, |10\rangle, |11\rangle$) for the circuit below. Can you describe the operation being performed?



7. Considering an n-qubit register of a quantum circuit, a single-qubit operator A over a qubit q of the system can be written as ¡ global operation:

$$\mathbf{Id_0} \otimes ... \otimes \mathbf{Id_{q-1}} \otimes A \otimes \mathbf{Id_{q+1}} \otimes ... \otimes \mathbf{Id_{n-1}}$$

where $\mathbf{Id_j}$ is an identity operator over qubit j.

Implement a function that takes a single-qubit operation, and a tuple (q, reg) containing the index of the target qubit, and the register size (i.e. number of qubits in the circuit), and generates the full quantum operation over the qubit register.

```
qc :: [[Complex Float]] -> (Int, Int) -> [[Complex Float]]
```

8. A matrix representing the CNOT gate in a quantum circuit cannot be obtained by application of tensor products if the control and target qubits are in non-consecutive registers. Below are three example cases of CNOT matrices:

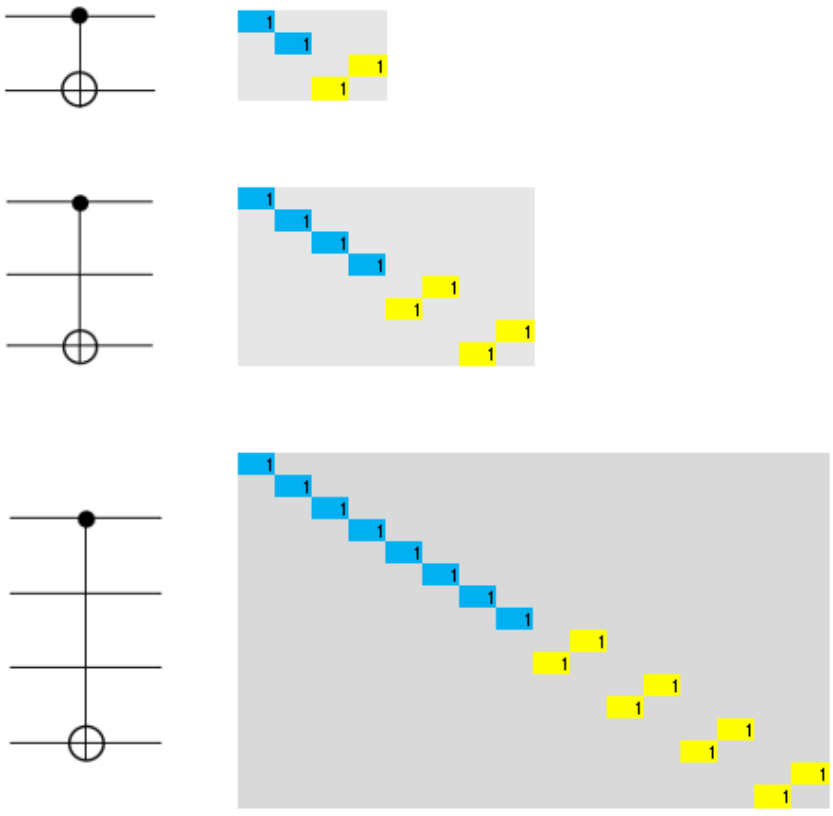


Figure 1: Different instances of CNOT gates in a quantum circuit. Entries sitting on the main diagonal are in blue, while entries outside of it are in yellow; gray space represents null entries.

Implement a function that takes an integer n representing the number of qubits over which the quantum gate is being performed, and returns a matrix in the form [[Complex Float]] representing the operation.

```
multi_cnot :: Int -> [[Complex Float]]
```

9. Implement a function that builds the full matrix of a global operation over a qubit register for an arbitrary CNOT:

```
qcnot :: (Int, Int, Int) -> [[Complex Float]]
```

The function takes a triple `(ctr, trg, reg)`, with `ctr` representing the index of the control qubit, `trg` being the index of the target qubit, and `reg` the number of qubits in the quantum circuit.

NOTE: Contrary to the function `mcnot` from Exercise 8, `qcnot` should take into account *upside-down* CNOT gates, i.e. where the target qubit is in a lower index than the control qubit. Remember that a CNOT can be inverted by applying the Hadamard gate in the control and target qubits, before and after the CNOT, as shown in Exercise 6.c).

For a better visualization of the problem, see the example case below. Use it to verify that your function works correctly.

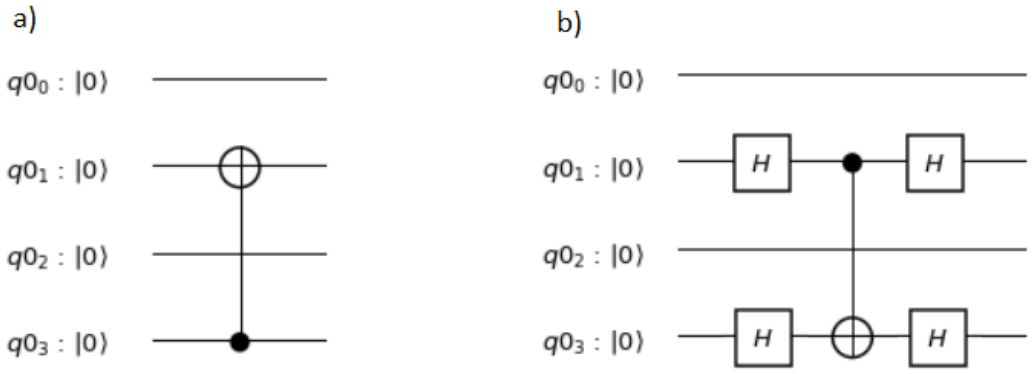


Figure 2: Circuits a) and b) are equivalent. This can be demonstrated through matrix multiplication, or using logic tables as in Exercise 6.c).

The circuit from figure 2.a) can be executed with `qcnot`:

```
*Main> disp_round $ qcnot (3,1,4)

[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
```

```
[0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0]
[0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0]
[0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0]
[0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0]
[0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
[0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0]
[0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0]
[0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0]
[0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0]
[0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0]
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1]
[0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0]
[0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0]
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0]
[0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0]
```