QHack

Quantum Coding Challenges

CHALLENGE COMPLETED

View successful submissions

Jump to code — Collapse text

Secrets in Spacetime

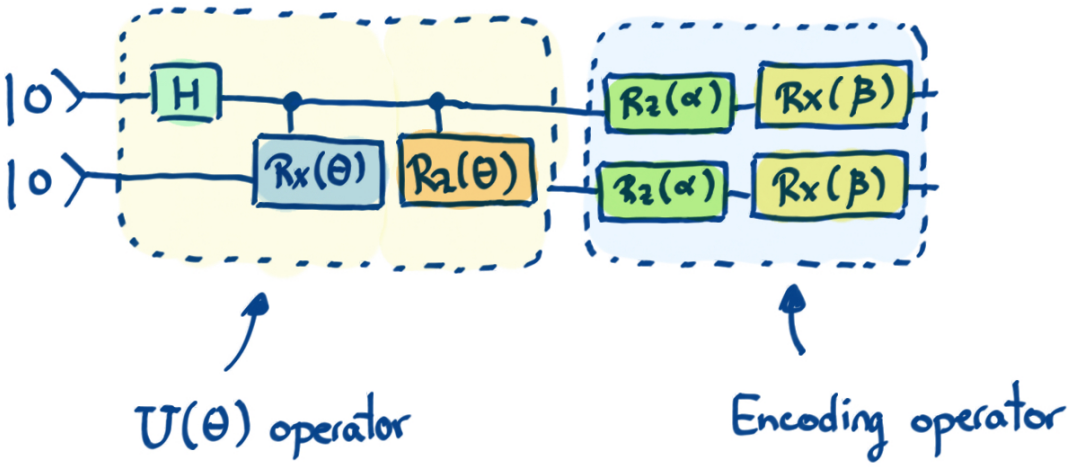
300 points

Backstory

Now Zenda and Reece have a cute way to send each other private messages using entangled qubits. Trine applauds them. "*Good work! But now that I think of it, superdense coding can be reversed, in a manner of speaking, to send quantum information using entanglement and classical bits. This will not only bring us to the last Law of Infodynamics, but teach us some basic facts about spacetime! Certain things have to be hidden from Nature itself.*" Zenda and Reece look perplexed. Trine smiles: "*Wait until I show you what timbits can do!*"

From causality to encryption

Zenda needs to send quantum states to Reece over a channel where someone could intercept the messages. They decide to encode the states they want to send with rotations on all of the qubits. To do this, they have chosen two real numbers, α and β, in advance, so that the states can be encoded as follows:



In this case, $U(\theta)$ is defined as the gate that generates the state $|\psi\rangle$ — what Zenda wants to send to Reece — that depends on a real number θ. Thus, if someone intercepts the message, instead of getting state $|\psi\rangle$ they will get state $(R_X(\beta)R_Z(\alpha))^{\otimes 2}|\psi\rangle$.

Although it seems like a super secure encoding procedure, it is not perfect! Once α and β have been chosen, there are certain values of θ that could make $(R_X(\beta)R_Z(\alpha))^{\otimes 2}|\psi\rangle = |\psi\rangle$ for certain states. This is a big problem — it means that someone is going to intercept the hidden state!

We will say that α and β are ϵ-unsafe values if there exists a θ such that

$$|\langle 0|U^\dagger(\theta)(R_X(\beta)R_Z(\alpha))^{\otimes 2}U(\theta)|0\rangle|^2 \geq 1-\epsilon.$$

Your goal is to determine if α and β are unsafe values given ϵ.

▸ **Laws of Infodynamics Part III: The Fourth Law**

Challenge code

In the code below, you are given a function called `is_unsafe`. **You must complete this function** by coming up with a way — you are given total freedom, from making a variational circuit to finding an analytical solution — to determine if the given values of α and β values are ϵ-unsafe.

Inputs

As input to this problem, you are given a `list(float)` containing the values of α, β, and ϵ, in that order.

Output

This code must output a boolean — `True` or `False` — corresponding to whether the values of α and β are ϵ-unsafe. For example, if you determine that the given values of α and β *aren't* ϵ-unsafe, your code must output `False`.

If your solution is correct, the output will be `"Correct!"` Otherwise, you will receive a `"Wrong answer"` prompt.

Good luck!

Code

Help

```
1  import json
2  import pennylane as qml
3  import pennylane.numpy as np
```

```python
def U_psi(theta):
    """
    Quantum function that generates |psi>, Zenda's state wants t

    Args:
        theta (float): Parameter that generates the state.

    """
    qml.Hadamard(wires = 0)
    qml.CRX(theta, wires = [0,1])
    qml.CRZ(theta, wires = [0,1])

def is_unsafe(alpha, beta, epsilon):
    """
    Boolean function that we will use to know if a set of parame

    Args:
        alpha (float): parameter used to encode the state.
        beta (float): parameter used to encode the state.
        epsilon (float): unsafe-tolerance.

    Returns:
        (bool): 'True' if alpha and beta are epsilon-unsafe coef

    """

```

```python
    # Put your code here #

```

```python
# These functions are responsible for testing the solution.
def run(test_case_input: str) -> str:
    ins = json.loads(test_case_input)
    output = is_unsafe(*ins)
    return str(output)

def check(solution_output: str, expected_output: str) -> None:

    def bool_to_int(string):
        if string == "True":
            return 1
        return 0

    solution_output = bool_to_int(solution_output)
    expected_output = bool_to_int(expected_output)
    assert solution_output == expected_output, "The solution is

```

```python
test_cases = [['[0.1, 0.2, 0.3]', 'True'], ['[1.1, 1.2, 0.3]',
```

```python
for i, (input_, expected_output) in enumerate(test_cases):
    print(f"Running test case {i} with input '{input_}'...")

    try:
        output = run(input_)

    except Exception as exc:
        print(f"Runtime Error. {exc}")

    else:
        if message := check(output, expected_output):
            print(f"Wrong Answer. Have: '{output}'. Want: '{expe
        else:
            print("Correct!")
```

Copy all

Submit

Open Notebook

Reset