CUGate's params getter does not comply with circuit's assign_parameters #XXXX

New issue

Open **ANONYMOUS** opened this issue on Nov 30, 2021 · 3 comments

ANONYMOUS commented on Nov 30, 2021 • edited

Contributor

Environment

- **Qiskit Terra version**: `0a10008`
- **Python version**: 3.8.10 64-bit
- **Operating system**: windows 10 64-bit

What is happening?

This issue is proposed based on the discussion under #7283, especially the insight of @**QISKIT DEV**.

`CUGate` is the controlled version of `UGate`, a generic single-qubit rotation gate with 3 Euler angles. Most controlled gates have the same parameters as the base gate, but `CUGate` has an extra one: the global phase `γ`. To cope with this, `CUGate` uses a `@property` decorator to define a getter for the property `params`. This getter function creates a new array and returns it. It leads to problems that following operations may be applied to this newly created array instead of the real parameters of the `CUGate` object. For example, that happens in the function `QuantumCircuit.assign_parameters`, which is demonstrated in the following section.

The failure of `assign_parameters` on `CUGate` leads to another problem. Currently, a line is missing below line 652 in qiskit/circuit/library/standard_gates/equivalence_library.py.

```
q = QuantumRegister(2, "q")
theta = Parameter("theta")
phi = Parameter("phi")
lam = Parameter("lam")
cu3_to_cu = QuantumCircuit(q)
cu3_to_cu.cu(theta, phi, lam, 0, 0, 1)
# should have another line:
# _sel.add_equivalence(CU3Gate(theta, phi, lam), cu3_to_cu)
```

If the `add_equivalence` function is added, the test test_equivalence_phase would report a `TypeError` exception. `QuantumCircuit.assign_parameters` is called during the test,

How can we reproduce the issue?

```
from qiskit.circuit import QuantumCircuit, Parameter
ps = [Parameter("t"), Parameter("p"), Parameter("l"), Parameter("g")]
qc = QuantumCircuit(2)
qc.cu(*ps, 0, 1)
qc.cu3(*ps[:3], 0, 1)
print(qc)
print(qc.assign_parameters(dict(zip(ps, [0.1, 0.2, 0.3, 0.4])), inplace=False))
```

The output is

```
q_0: ──────■─────────────■──────
     ┌─────┴──────┐┌─────┴─────┐
q_1: ┤ U(t,p,l,g) ├┤ U3(t,p,l) ├
     └────────────┘└───────────┘

q_0: ──────■────────────────■─────────
     ┌─────┴──────┐┌────────┴────────┐
q_1: ┤ U(t,p,l,g) ├┤ U3(0.1,0.2,0.3) ├
     └────────────┘└─────────────────┘
```

suggesting the assignment of CUGate is failed. Note that the parameters of the `U3Gate` have been assigned.

What should happen?

The second circuit should be

```
q_0: ──────────■────────────────────■─────────
     ┌─────────┴──────────┐┌────────┴────────┐
q_1: ┤ U(0.1,0.2,0.3,0.4) ├┤ U3(0.1,0.2,0.3) ├
     └────────────────────┘└─────────────────┘
```

Assignees

No one assigned

Labels

bug

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

Notifications Customize

Subscribe

You're not receiving notifications from this thread.

3 participants

Any suggestions?

Maybe remove the `γ` parameter from CUGate. The global phase is a property of quantum circuits. It is not a property of other gates. Adding this parameter to CUGate makes it different from the general cases and requires special processing, e.g., here and here.

ANONYMOUS added the bug label on Nov 30, 2021

Cryoris commented on Nov 30, 2021 Contributor

We added the `gamma` parameter to the `CUGate` so that it represents a general 2 qubit gate, as discussed in #4106 Maybe another solution would be to just return the `params` stored in `CUGate` insted of going via the base gate which doesn't have the `gamma` parameter.

QISKIT DEV commented on Nov 30, 2021 • edited Contributor

Perhaps calling it "global" phase is a bit misleading here (it's the phase of the controlled gate), but yeah, the gamma parameter needs to stay for now. The main issue is [again :(] that we don't have a contract for what `Instruction.params` should contain, and how it should behave with respect to mutability. `QuantumCircuit` assumes that it's a "true" attribute, and tries to write to it as if it is guaranteed to be a list, but that's probably not something it can assume.

The "bug" could either be that `CUGate` *must* return a list-like object that backs *all* its parameters (in which case we need to do what Julien said), or other places in Terra need to stop assuming that they can mutate the object. They can write wholesale, perhaps (so they'll trigger the setter method), but they can't mutate. This second form is more safe, but it likely has some performance implications, because we'd need to be reconstructing several lists/tuples each time.

1

Cryoris commented on Dec 1, 2021 Contributor

Yep agreed, after #7087 we can hopefully start introducing more structure and determine which gates share which properties (-> e.g. standard gates that have free parameters).

Write Preview

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Comment

ⓘ Remember, contributions to this repository should follow its contributing guidelines and code of conduct.