isQ 编程语言用户手册

概述

isQ 是由中国科学院软件研究所开发的量子程序设计语言，目前开发团队正与中科弧光量子软件公司共同研发 isQ 新功能。开发团队实现了多个版本的编译器以将 isQ 对接到不同的硬件平台或模拟器上。本用户手册将详细介绍中科院量子云平台上所支持的 isQ 编程语言的使用方法——为了适配超导量子硬件的一些底层限制，我们对完整版 isQ 语言的语法特性做了一些简化，以使得用本手册中的 isQ 语言所编写的量子程序，能真正地运行在支持 QCIS 指令集的硬件平台上。接下来，我们将用 isQ-core 来表示云平台上所支持的 isQ 版本，以区别于完整的 isQ 语言。后续随着 QCIS 指令集和云平台的扩展，isQ-core 的语法也会进行迭代与更新。

下面，我们将首先给出量子云平台 isQ-core 的语法说明，然后再列举若干例子以便读者有更直观的理解。

量子云平台 isQ-core 语法说明

isQ-core 编程语言包含如下四种操作：*自定义 qubit 变量，酉操作，测量操作，for 循环*。完整的代码结构如下：

$$Qubit\ Define$$

…

$$procedure\ main()\{procedureBody\}$$

其中 procedureBody 由酉操作，测量操作以及 for 循环构成

自定义 qbit

isQ-core 支持用户定义 qubit 变量（仅支持定义全局变量），可为单个变量，或者 qubit 数组（注意在 isQ 中， qubit 对应的关键字是 “qbit”），定义格式如下：

$$qbit\ ID;$$

$$qbit\ ID[NUMBER];$$

$$qbit\ ID_1, \dots, ID_n[NUMBER];$$

其中 ID 为变量名，由字母和数字构成，如 a, b2，变量名不能重复；数组变量形式为 ID[NUMBER]，NUMBER 为数字，如 a[3]，b1[10]；每一行可定义单个或多个变量/数组。所有的 qubit 变量均默认初始化为|0>

qbundle

在说明 procedureBody 之前，我们先介绍一个概念：qbundle。qbundle 表示 qubit 数组内的一组 qubit，其表达形式如下：

$$ID[idx_1, idx_2, \dots, idx_n]$$

$$ID[start: stop: step]$$

Qbundle 有两种表达形式，其中 ID 为 qubit 变量名，必须为之前定义的数组变量。在第一种表达形式中，用一系列下标来表示这一组 qubit，如我们之前定义了 t[10]，则可以用 t[1,3,5,7]来表示 t[1],t[3],t[5],t[7]这四个 qubit。第二种表达形式类似 python 的 range，用 start/stop 表示起始/终止下标（不包含 stop），step 表示步长，默认为 1。如 t[1:4]表示 t[1],t[2],t[3]这三个 qubit，t[1:6:2]表示 t[1],t[3],t[5]这三个 qubit

酉操作

酉操作为对 qubit 作用酉门，定义格式如下：

$$GateID < qid_1, \dots, qid_n >$$

$$GateID < qbundle_1, \dots, qbundle_n >$$

GateID 为酉门名称，现有基础门包括：H，X，Y，Z，S，T，SD，TD，X2P，X2M，Y2P，Y2M， CZ，n 为门作用的 qubit 数目（CZ 为 2，其余门均为 1）。酉门可作用在 qubit 变量上（形式一）或者 qbundle（形式二）上。当作用在 qbundle 上

时，$qbound_i$的长度必须一致，此时其语义为对每个 j，执行

$$GateID < qbundle_{1,j}, \dots, qbundle_{n,j} >$$

如 H<t[1:3]>表示

H<t[1]>;

H<t[2]>;

CNOT<t[1:3], w[2,4]>表示

CNOT<t[1], w[2]>;

CNOT<t[2], w[4]>;

需要注意的是：CZ 需要作用的相邻的 qubit 上（根据定义 qubit 的先后顺序判断是否相邻)；已测量完的 qubit 变量不可再执行酉操作。

测量操作

对 qubit 进行测量，定义格式类似上面的酉操作

$$M < qid >$$

$$M < qbundle >$$

其中对 qbundle 测量的语义为对 qbundle 里每一个 qubit 进行测量

for 循环

为方便用户做一些循环酉操作，简化代码，isQ-core 提供了 for 循环的支持，定义格式如下：

$$for\ cid\ in\ start{:}\ stop{:}\ step\{$$

$$forloopBody$$

$$\}$$

其中 cid 为 for 循环自增变量名，start，stop，step 和 qbundle 中定义一致，

start/stop 表示起始/终止值（不包含 stop），step 表示步长。forloopBody 定义和 procedureBody 一致，里面可包含酉操作，测量操作以及 for 循环。值得注意的是在 forloopBody 中可用带 cid 的表达式做下标来访问 qubit 变量，如：

$$for\ i\ in\ 1{:}3\{$$

$$H < t[i] >;$$

$$CZ < w[i], w[i+1] >;$$

$$\}$$

注释

isQ-core 提供单行注释功能，在需要注释内容前加 // 即可

isQ-core 样例：

本节将介绍若干 isQ-core 样例，以供用户参考。这些样例均能在 isQ-QCIS 编译器上编译通过。

样例一

```
qbit p,q,r;
qbit w[5];

procedure main() {

    H<w[0:5]>;
    M<w[1,3,4]>;
}
```

在该代码中，我们定义了 8 个 qubit，其中三个单变量 p,q,r，一个数组变量 w[5]。在 main 函数中，我们对 w 数组内的每一个 qubit 做了一次 H 门，然后测量了 w[1],w[3],w[4]三个 qubit

样例二

CZ 门用法（目前超导芯片只支持最多 12 个 qubit，且为线性排列，只能对相邻的 qubit 做 CZ 门）

```
qbit p,q;
qbit w[5];

procedure main() {

    CZ<p, q>;
    CZ<q, w[0]>;
    CZ<w[1:3], w[2:4]>;
    M<w[0:5]>;
    // w[1], w[3] is not adjacent, can't do CZ
    // CZ <w[1], w[3]>; compile failed
}
```

在该代码中，我们对（p，q），（q，w[0]），（w[1]，w[2]），（w[2]，w[3]）做了 CZ 门。上述 4 组 qubit 都是物理上相邻的 qubit，所以可以做 CZ。而注释中的（w[1]，w[3]）不相邻，因此不能做 CZ，否则会报编译错误。

样例三

for 循环简单用法

```
qbit w[5];

procedure main() {

    for i in 0:3{
        X2P<w[i]>;
        Y2P<w[i+1]>;
        CZ<w[i], w[i+1]>;
    }

    for i in 0:3{
```

```
        for j in 0:3{
            for k in 0:3{
                H<w[(i+j+k) % 5]>;
            }
        }
    }

    M<w[0:4]>;
}
```

在该代码中，我们使用 isQ-core 的 for 循环功能，其中第一个 for 循环 i 做了三次迭代（0，1，2 三个值），每次迭代均对 w[i]，w[i+1]两个 qubit 做相应的门操作。第二个 for 由三个 for 循环嵌套构成，可以看到 isQ-core 也可以通过简单的四则运算来获取下标。