Contents

1 QLisp 核心语法与概念指南 **1**

1.1 目录 1
1.2 核心语法 2
1.3 电路表示与参数化 2
1.3.1 基本语法示例 2
1.3.2 S21 实验线路 (实际案例) 3
1.4 量子门列表 3
1.4.1 单比特门 3
1.4.2 双比特门 3
1.5 预定义电路 4
1.5.1 量子过程层析 (QPT) 4
1.5.2 其他序列 4
1.6 工具函数 4
1.7 示例 4
1.7.1 示例 1：Bell 态制备和测量 4
1.7.2 示例 2：参数化的 Rabi 实验 5
1.8 最佳实践 6

1 QLisp 核心语法与概念指南

QLisp 是 QuarkStudio 项目中用于描述和操作量子计算电路的规范。本指南专注于讲解在 `Recipe` 系统中实际使用的 **核心语法**，这对于编写参数化的物理实验至关重要。

> **注意**: QLisp 可能存在一个更高层次、更简洁的抽象语法用于教学或快速构建，但本指南描述的是与后端编译器直接交互的、功能最强大的核心格式。

1.1 目录

- 核心语法
- 电路表示与参数化
- 量子门列表
- 预定义电路
- 工具函数
- 示例

1.2 核心语法

在 Recipe 系统中，量子电路由一个 **Python 列表（List）**表示。列表中的每一个元素都是一个二元元组 (operation, target)，代表一个量子操作。

```
[ (operation_1, target_1), (operation_2, target_2), ... ]
```

- **target**: 应用该操作的目标量子比特，可以是单个比特 ('Q0') 或多个比特 ('Q0', 'Q1')。
- **operation**: 这本身也是一个元组，定义了操作的名称和传递给它的参数。

1.2.0.1 operation 元组的结构

operation 元组的格式为 ('GateName', param1, param2, ...)。

- **'GateName' (字符串, 必需)**: 操作的名称。这个字符串会映射到后端门库中定义的一个具体物理实现 (即脉冲序列)。例如：'X90', 'CZ', 'Measure'。
- **param1, param2, ... (可选)**: 传递给该操作的参数。这些参数可以在 Recipe 中被动态赋值或扫描。

示例: – 一个无参数的 X90 门作用于 Q1: (('X90',), 'Q1') – 一个 CZ 门作用于 Q1 和 Q2: (('CZ',), ('Q1', 'Q2')) – 一个带参数的测量操作: (('Measure', 0), 'Q0') (这里的 0 是一个传递进去的参数)

1.3 电路表示与参数化

在实际实验中，我们通常将线路编写为一个 Python 函数。函数的参数（如 qubits, freq, amp 等）由 Recipe 在执行时动态传入。

1.3.1 基本语法示例

```python
# 线路函数接收 Recipe 传入的参数
def my_circuit(qubits: list[str], amp: float, ctx=None):
    # 第一个参数 'qubits' 是一个包含所有目标比特名称的列表
    q0 = qubits[0]
    q1 = qubits[1]

    circuit = [
        # 对 q0 应用一个 X90 门
        (('X90',), q0),
        # 对 q1 应用一个带可变幅度的 Y 门
        # 'amp' 的值由 Recipe 在运行时提供
        (('Y', amp), q1),
        # 执行 CZ 门
        (('CZ',), (q0, q1)),
        # 执行测量
```

```
        (('Measure', 0), q0),
        (('Measure', 1), q1)
    ]
    return circuit
```

1.3.2 S21 实验线路（实际案例）

这是 s21.py 中使用的线路，完美地展示了如何通过 enumerate 将索引 i 作为参数传递给测量操作。

```
def s21_circuit(qubits: list[str], freq: float, ctx=None):
    """
    为列表中的每个量子比特生成一个测量操作。
    freq 参数虽然在此处定义，但其值由 Recipe 在外部控制。
    """
    # (('Measure', 0), 'Q0'), (('Measure', 1), 'Q1'), ...
    cc = [(('Measure', i), q) for i, q in enumerate(qubits)]
    return cc
```

1.4 量子门列表

下表列出了一些常用的门操作名称（'GateName'）。实际可用的门由后端门库定义。

1.4.1 单比特门

门名称	描述
'X' or 'X180'	Pauli-X 门（π 脉冲）
'X90'	π/2 X-脉冲
'Y' or 'Y180'	Pauli-Y 门（π 脉冲）
'Y90'	π/2 Y-脉冲
'H'	Hadamard 门
'Z'	Pauli-Z 门（软件实现的相位门）
'Measure'	测量操作

1.4.2 双比特门

门名称	描述
'CZ'	受控 Z 门
'CX'	受控 X 门（CNOT）

门名称	描述
'iSWAP'	iSWAP 门

1.5 预定义电路

QLisp 提供了多种预定义的电路模板，这些是更高层次的封装，便于快速调用标准实验序列：

1.5.1 量子过程层析 (QPT)

```
import qlisp.circuits as qc

qpt_circuit = qc.qpt(qubits=['Q0', 'Q1'])
```

1.5.2 其他序列

```
# Ramsey 序列
ramsey_circuit = qc.Ramsey(qubit='Q0')

# Spin Echo 序列
spinecho_circuit = qc.SpinEcho(qubit='Q0')
```

1.6 工具函数

函数	描述
draw(circuit)	绘制电路图
seq2mat(circuit)	将电路转换为矩阵
applySeq(circuit, state)	将电路应用于量子态

1.7 示例

1.7.1 示例 1：Bell 态制备和测量

```
def bell_state_circuit(qubits: list[str], ctx=None):
    """ 制备并测量 Bell 态 |Φ+ """
```

```
    q_control = qubits[0]
    q_target = qubits[1]

    # 步骤 1: 制备 Bell 态
    bell_prep = [
        (('H',), q_control),
        (('CX',), (q_control, q_target)),
    ]

    # 步骤 2: 测量
    measurement = [
        (('Measure', 0), q_control),
        (('Measure', 1), q_target),
    ]

    return bell_prep + measurement

# 在 Recipe 中使用:
# rcp.circuit = bell_state_circuit
# rcp['qubits'] = ('Q0', 'Q1')
```

1.7.2 示例 2：参数化的 Rabi 实验

```
def rabi_circuit(qubits: list[str], pulse_amp: float, ctx=None):
    """
    执行 Rabi 实验的线路。
    对一个比特施加一个可变幅度的 X 脉冲，然后进行测量。
    """
    target_qubit = qubits[0]

    # 脉冲的幅度 `pulse_amp` 将由 Recipe 在扫描时动态提供
    rabi_pulse = [
        (('X', pulse_amp), target_qubit)
    ]

    measurement = [
        (('Measure', 0), target_qubit)
    ]

    return rabi_pulse + measurement

# 在 Recipe 中使用:
# rcp.circuit = rabi_circuit
# rcp['qubits'] = ('Q3',)
# rcp['pulse_amp'] = np.linspace(0, 1.0, 101) # 扫描幅度从 0 到 1
```

1.8 最佳实践

1. **命名规范**: 使用有意义的量子比特名称（如'Q0', 'Q1'）。
2. **电路结构**: 将复杂电路分解为子电路，或使用函数生成。
3. **注释**: 为复杂的量子操作或线路函数添加清晰的文档字符串。
4. **参数化**: 充分利用 `Recipe` 系统，将需要扫描或改变的物理量（幅度、频率、时间）作为线路函数的参数。

本指南基于 QLisp 库的当前版本编写，如有更新请参考最新文档。