Quantum Controlled Measurement via Program Transformation

(work-in-progress)

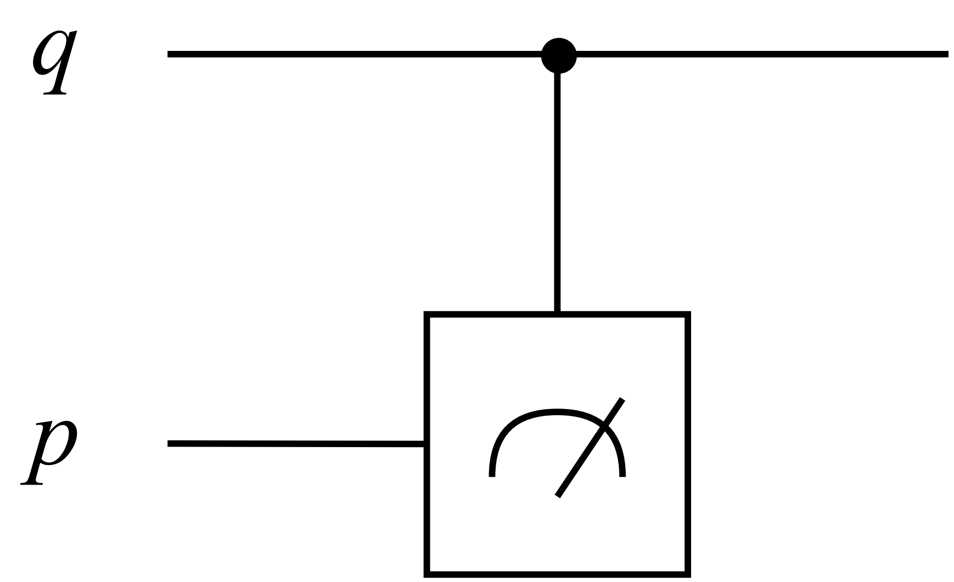
PLanQC 2023

Kengo Hirata (Edinburgh University, Kyoto University)
Takeshi Tsukada (Chiba University)

This talk

- About the semantics of programs like

```
// p, q: qbit
qif q {
    p ← meas(p)
} else {
    nope
}
```



Motivating Example

Quantum SWITCH

qif x { $EF(y)$ } else { $FE(y)$ }

E, F : quantum channel

cannot be written in most of the languages.

qif q { $U(p)$ } else { ..

Only unitaries

Problem: semantics for quantum controlled channel is ill-defined in general.

Previous Work and Question

QuGCL [Ying 16] has “qif” & measurement (without limitation)

- ✘ lacks physical implementation
- ? some ambiguities in semantics

Q. Can we define a clearer semantics with simple physical implementation for such programs?

qif q {
    $p \leftarrow$ meas(p)
} else {
    nope
}

Contribution

Proposal: <u>Program Transformation Technique</u>

- A program with measurement inside “qif”

Deferring measurement

- A program without them.

Result: A simple <u>physical implementation</u>

Application: <u>Quantum-controlled while loop</u>

Outline

- Program transformation
- Comparison: Equality of semantics
- Application: Quantum controlled while loop

Program Transformation

Language & Example

Types

T ::= qbit | bool

linearly used

Example

(Rough Idea)

qif q {
  $p \leftarrow$ meas(p)
} else {
  nope
}

(Well formed)

qif q {
  $b \leftarrow$ meas(p_0)
  if b {
    $p_1 \leftarrow |1\rangle$
  } else {
    $p_1 \leftarrow |0\rangle$
  }
} else {
  $p_1 \leftarrow p_0$
}

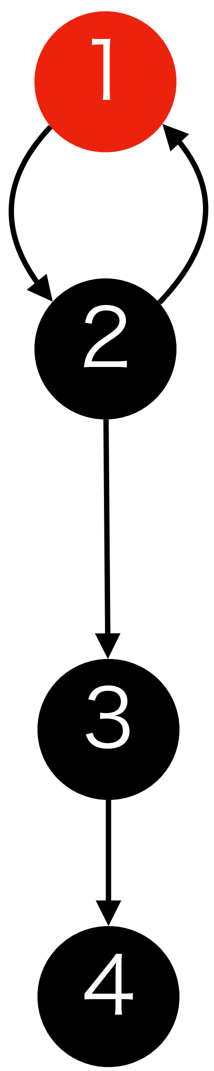
Program Transformation

Algorithm Flow

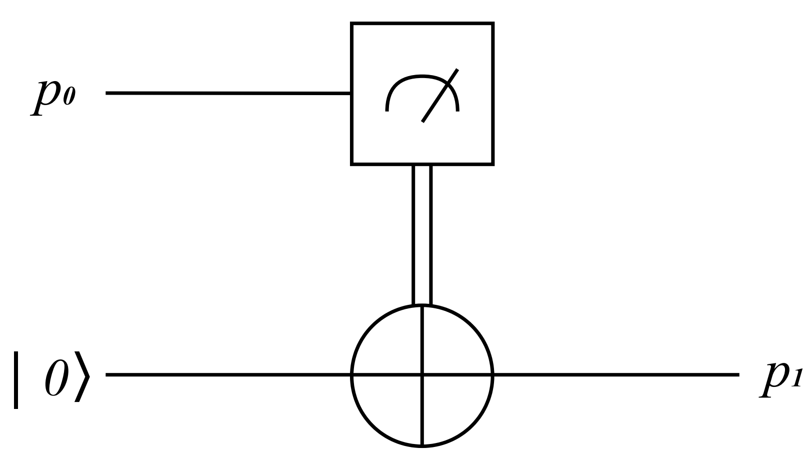
1. Deferring measurements (inside of “qif”)
2. Deferring measurements (outside of “qif”)
3. Introduction of dummy value
4. Arrangement

Program Transformation

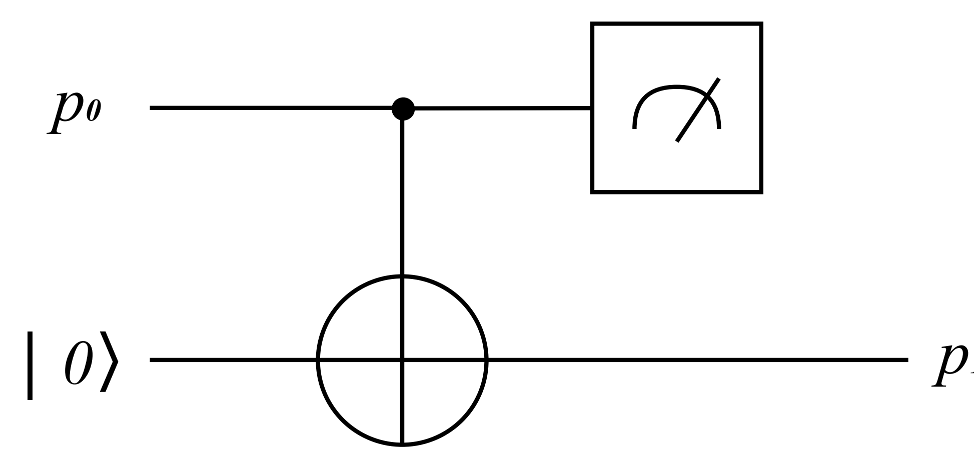
1. Deferring measurements (inside of "qif")



qif q {
    $b \leftarrow \mathrm{meas}(p_0)$
    if b {
        $p_1 \leftarrow |1\rangle$
    } else {
        $p_1 \leftarrow |0\rangle$
    }
} else {
    $p_1 \leftarrow p_0$
}

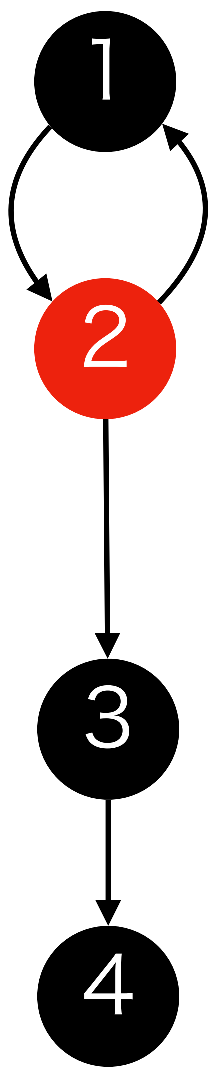


qif q {
    qif p_0 {
        $p_1 \leftarrow |1\rangle$
    } else {
        $p_1 \leftarrow |0\rangle$
    }
    $b \leftarrow \mathrm{meas}(p_0)$
} else {
    $p_1 \leftarrow p_0$
}



Program Transformation

2. Deferring measurements (outside of “qif”)

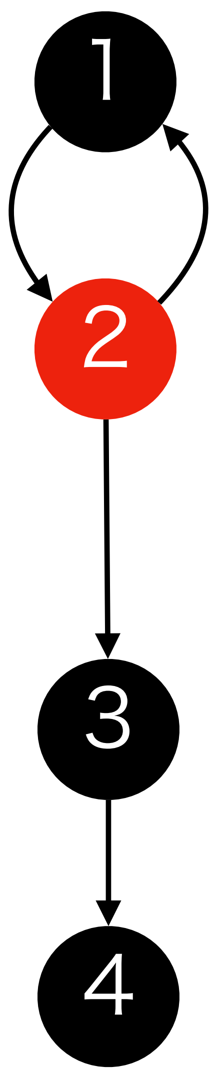


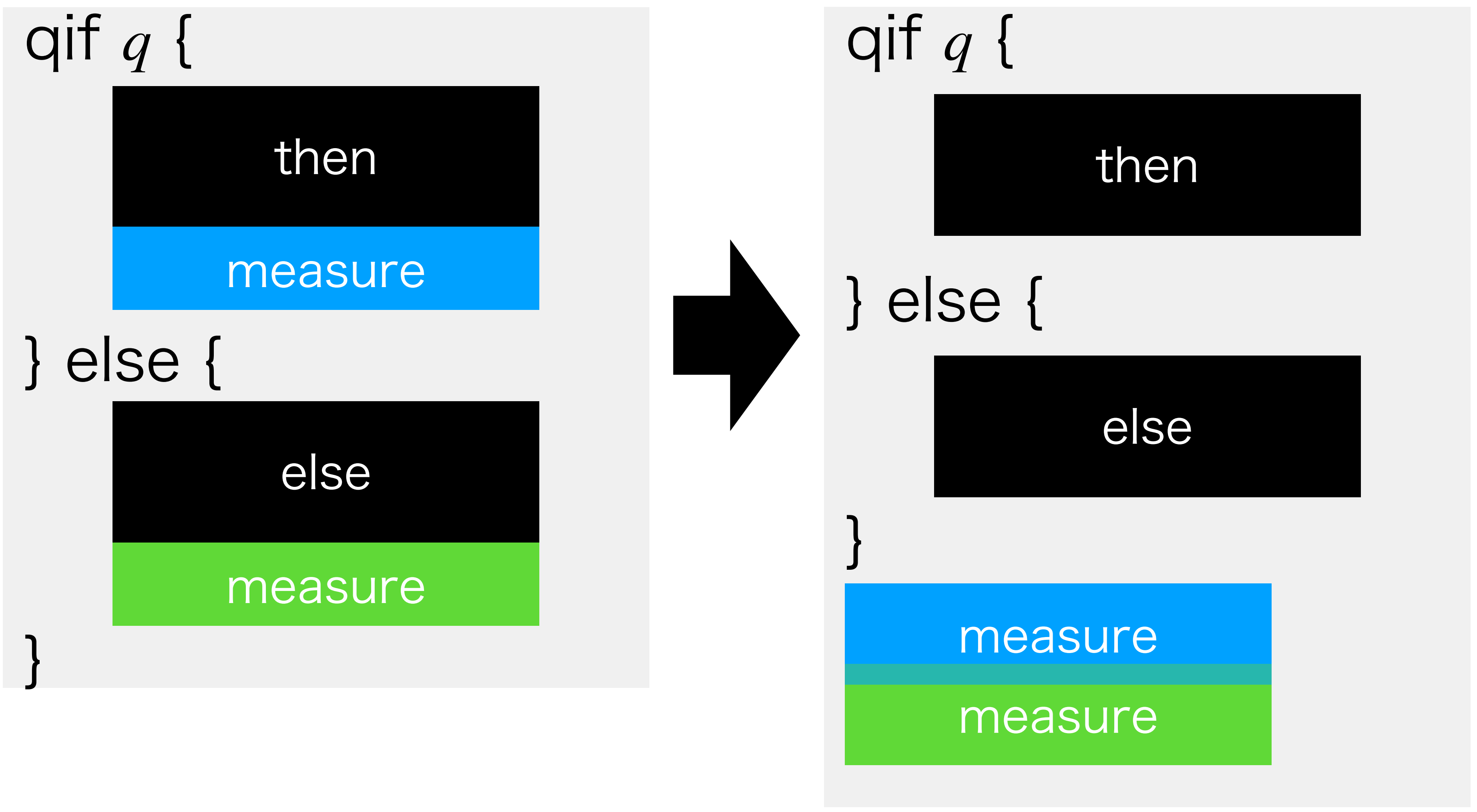
qif q {
    qif p_0 {
        $p_1 \leftarrow |1\rangle$
    } else {
        $p_1 \leftarrow |0\rangle$
    }
    $b \leftarrow$ meas(p_0)
} else {
    $p_1 \leftarrow p_0$
}

qif q {
    qif p_0 {
        $p_1 \leftarrow |1\rangle$
    } else {
        $p_1 \leftarrow |0\rangle$
    }
} else {
    $p_1 \leftarrow p_0$
}
$b \leftarrow$ meas(p_0)

Program Transformation

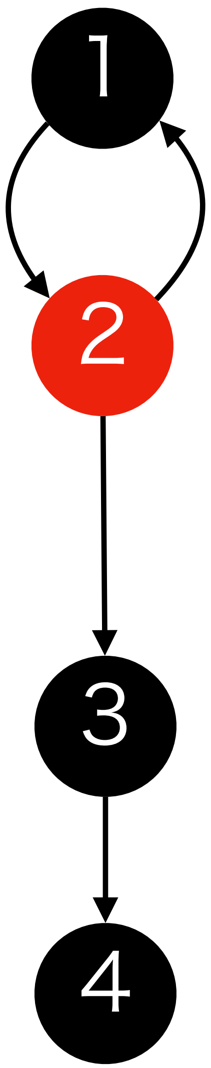
2. Deferring measurements (outside of “qif”)





Program Transformation

2. Deferring measurements (outside of “qif”)



Type Mismatch (violate the linearity)

qif q {
    qif p_0 {
        $p_1 \leftarrow |1\rangle$
    } else {
        $p_1 \leftarrow |0\rangle$
    }
} else {
    $p_1 \leftarrow p_0$
}
$b \leftarrow$ meas(p_0)

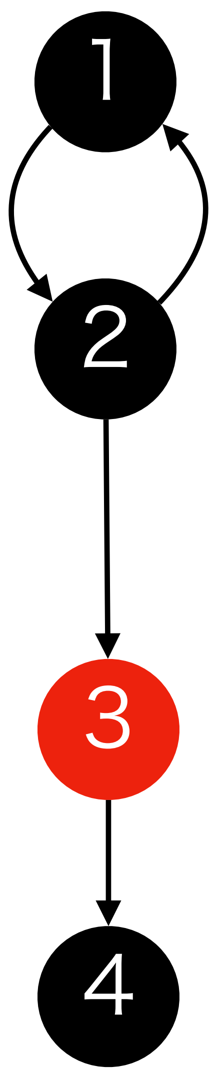
p_0 alive

$\nparallel$

p_0 consumed

Program Transformation

3. Introduction of dummy value



qif q {
    qif p_0 {
        $p_1 \leftarrow |1\rangle$
    } else {
        $p_1 \leftarrow |0\rangle$
    }
} else {
    $p_1 \leftarrow p_0$

p_0: consumed

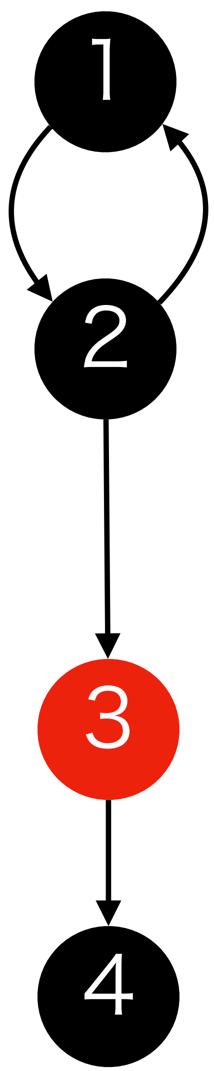
}
$b \leftarrow$ meas(p_0)

p_0: used again

qif q {
    qif p_0 {
        $p_1 \leftarrow |1\rangle$
    } else {
        $p_1 \leftarrow |0\rangle$
    }
} else {
    $p_1 \leftarrow p_0$
    $\mathbf{p_0} \leftarrow$ dummy
}
$b \leftarrow$ meas(p_0)

Program Transformation

3. Introduction of dummy value



qif q {
    qif p_0 {
        $p_1 \leftarrow |1\rangle$
    } else {
        $p_1 \leftarrow |0\rangle$
    }
} else {
    $p_1 \leftarrow p_0$
}
$b \leftarrow$ meas(p_0)

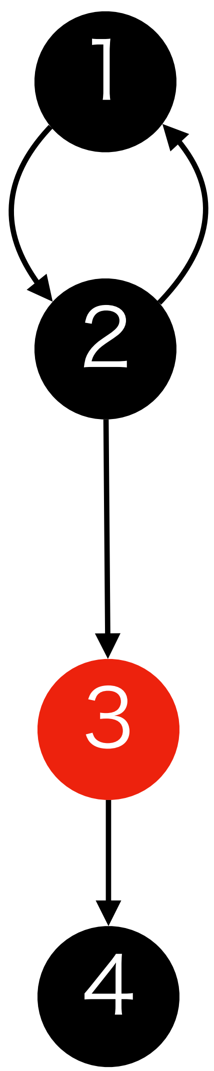
p_0: consumed

Choose $|0\rangle$

qif q {
    qif p_0 {
        $p_1 \leftarrow |1\rangle$
    } else {
        $p_1 \leftarrow |0\rangle$
    }
} else {
    $p_1 \leftarrow p_0$
    $p_0 \leftarrow |0\rangle$
}
$b \leftarrow$ meas(p_0)

Program Transformation

3. Introduction of dummy value



qif q {
  qif p_0 {
    $p_1 \leftarrow |1\rangle$
  } else {
    $p_1 \leftarrow |0\rangle$
  }
} else {
  $p_1 \leftarrow p_0$
}
$b \leftarrow \mathrm{meas}(p_0)$

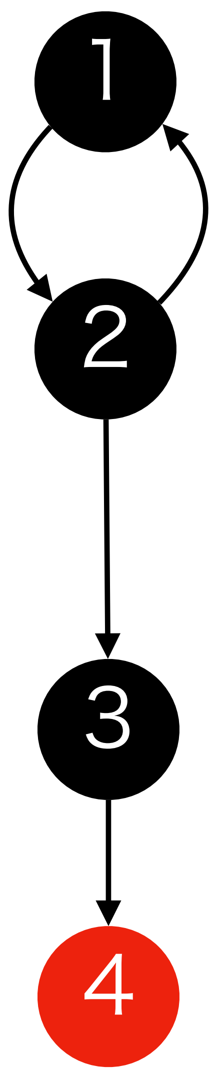
p_0: consumed

Choose $|+\rangle$

qif q {
  qif p_0 {
    $p_1 \leftarrow |1\rangle$
  } else {
    $p_1 \leftarrow |0\rangle$
  }
} else {
  $p_1 \leftarrow p_0$
  $p_0 \leftarrow |+\rangle$
}
$b \leftarrow \mathrm{meas}(p_0)$

Program Transformation

4. Arrangement



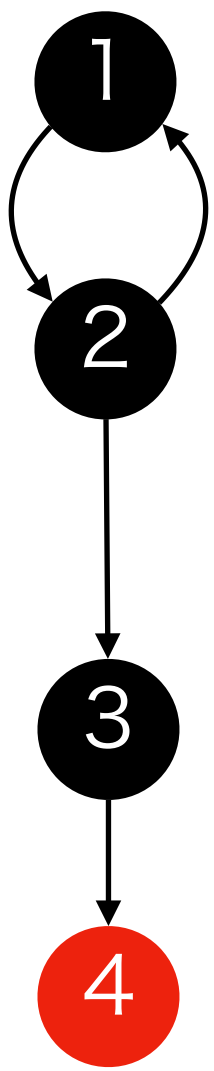
qif q {
    qif p_0 {
        $p_1 \leftarrow |1\rangle$
    } else {
        $p_1 \leftarrow |0\rangle$
    }
} else {
    $p_1 \leftarrow p_0$
    $p_0 \leftarrow |+\rangle$
}
$b \leftarrow$ meas(p_0)

Controlled isometry

Measurement

Program Transformation

4. Arrangement



$p_1 \leftarrow |+\rangle$

qif q {

  $p_1 \leftarrow \mathrm{CH}(p_1)$

  $p_0, p_1 \leftarrow \mathrm{CNot}(p_0, p_1)$

} else {

  $p_0, p_1 \leftarrow \mathrm{swap}(p_0, p_1)$

}

$b \leftarrow \mathrm{meas}(p_0)$

Initial State

Controlled Unitary

Measurement

Program Transformation

Algorithm Flow

1. Deferring measurements (inside of “qif”)
2. Deferring measurements (outside of “qif”)
3. Introduction of dummy value
4. Arrangement

Outline

- Program transformation
- Comparison: Equality of semantics
- Application: Quantum controlled while loop

Equality of semantics

Quantum switch

Theorem

Our semantics via program transformation defines the correct semantics on Quantum SWITCH.

Equality of semantics

Comparison with Ying's semantics for QuGCL

Ying defined two semantics for QuGCL.

1. Original semantics

2. General semantics determined for each parameter
(includes 1. as an instance)

Theorem

In the simplest example we studied above,
"the choice of dummy value"
= "parameter for general semantics of QuGCL".

Equality of semantics

Comparison with Abbott et al. 20

[Abbott et al. 20]

To define a quantum controlled channel, we need not only the Kraus decomposition of channels, but <u>additional information of "initial states of environment"</u>.

<u>Observation</u>

The "initial states of environment" is closely related to our "dummy value".

➡ Found non-trivial connection between [Ying 16] and [Abbott et al. 20]

Outline

- Program transformation
- Comparison: Equality of semantics
- Application: Quantum controlled while loop

Quantum controlled while loop

Background

```
// d : Data,  q : qbit
qwhile q {
    (d, q) ← M(d)
}
```

	Semantics	Physical implementation	Measurement in M
Bădescu et al. 15	No		
Ying et al. 14	Yes (Fock space)	No	Yes
Sabry et al. 18	Yes (List of qubits)	Yes	No

Quantum controlled while loop

Question

- [Sabry et al. 18] Semantics with list of qubits

  Has implementation M has to be unitary

Q. Can we remove this limitation by moving measurements outside of “qwhile”?

A. Yes!

Quantum-while with measurement

$\longrightarrow$ Sabry et al.’s language

= Quantum-recursion + list of qubit

Quantum controlled while loop

Program Transformation

Source program

```
// d : Data,  q : qbit
qwhile q {
    (d, q) ← M(d)
}
```

First step: Apply the program transformation to M

$$\mathrm{M} \quad \mapsto \quad \begin{array}{l} \left.\begin{array}{l} q \leftarrow |0\rangle \\ a \leftarrow |0\cdots0\rangle \end{array}\right\} \text{Initial state} \\ \left.(d,\ q,\ a) \leftarrow \overline{\mathrm{M}}(d,\ q,\ a)\right\} \text{Unitary} \\ \left.\mathrm{meas}(a)\right\} \text{Meas.} \end{array}$$

$\overline{\mathrm{M}}$: unitary part of M

Quantum controlled while loop

Program Transformation

Source program

```
// d : Data,  q : qbit
qwhile q {
    (d, q) ← M(d)
}
```

First step: Apply the program transformation to M

$$\mathrm{M} \quad \mapsto \quad \begin{array}{l} q_i \leftarrow |0\rangle \\ a_i \leftarrow |0\cdots0\rangle \\ (d, q_i, a_i) \leftarrow \overline{\mathrm{M}}(d, q_i, a_i) \\ \mathrm{meas}(a_i) \end{array}$$

Prepare "list of qs and as",

and apply $\overline{\mathrm{M}}$ at the i-th iteration

Quantum controlled while loop

Program Transformation

Second step: rewrite qwhile

Idea:

lq : list of q

la : list of a

helper function

main part

```
// d : Data,  q : qbit,  lq : [qbit],  la : [qbit],
fun W(d, q, lq, la) {
    qif q {
        if let (q′ :: lq′, a′ :: la′) = (lq, la) {      ← new_q = lq.next()
            (d, q′, a′) ← M̄(d, q′, a′)                   ← Apply M̄
            (d, lq′, la′) ← W(d, q′, lq′, la′)            ← Recursion
        }
    }
    (d, q :: lq′, a′ :: la′)
}
(d, lq, la) ← W(d, q, |0⋯0⟩, |0⋯0⟩)
meas(la)
```

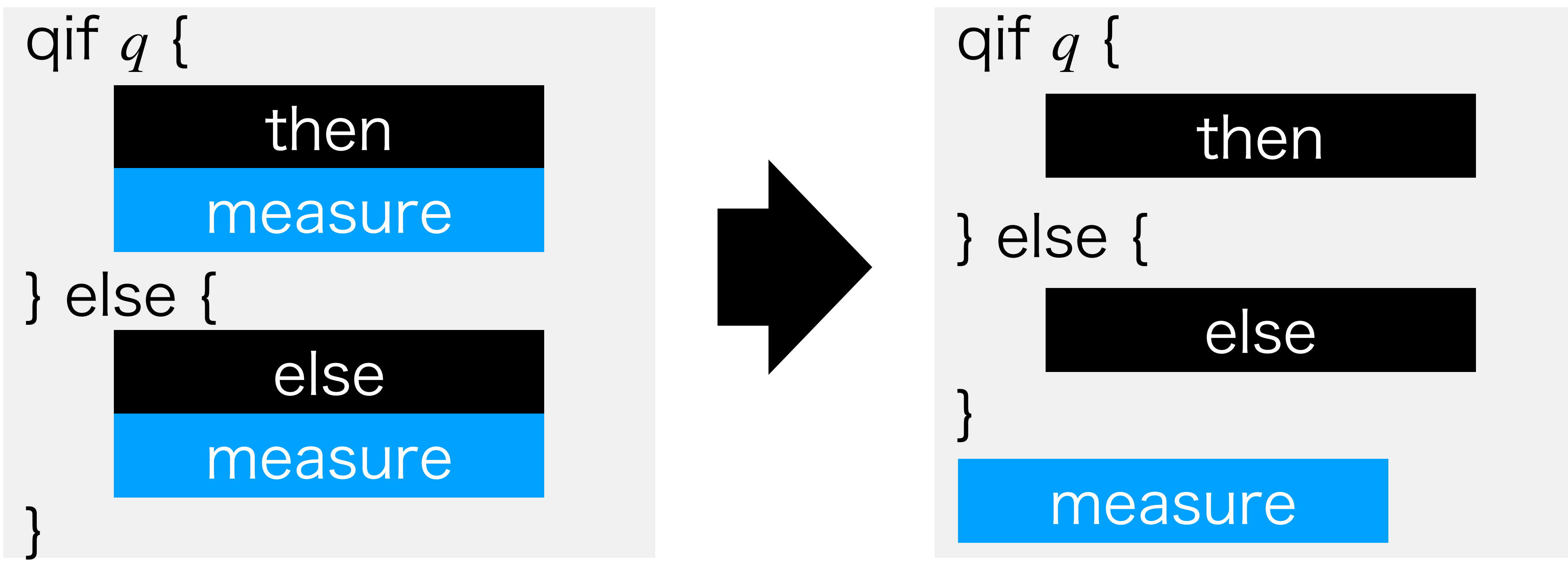
Conclusion

- We propose a program transformation technique which moves measurement out of “qif” statement.
  - Physical implementable semantics.
  - Found non-trivial connection between Ying’s work and Abbott et al’s work.
  - This technique is also applicable to “qwhile”.

Future Work

- In some cases, we do not need to introduce dummy value.

  (e.g. Quantum SWITCH)



- And in some cases, quantum control on CPTP maps is well-defined [Abbott et al. 20].

  (e.g. Quantum SWITCH)

(when “then” and “else” branches uses the same combination of CPTP maps)