LinguaQuanta: Initial Results on a Quantum Transpiler

Scott Wesley
Dalhousie University

Outline of Talk

1. **Motivations and Background**
2. **Correctness**: Categorical Specifications
3. **Translations**: Some Decompositions
4. **Challenges**: Ancilla Management

Acknowledgements

Colleagues and Collaborators

Key Collaborators

Dr. Peter Selinger (Dalhousie University)

Provided early consultation of semantics.

Xiaoning Bian (Dalhousie University)

Initiated the LinguaQuanta project.

Additional Project Support

Dr. Neil J. Ross (Dalhousie University)

Provided feedback as my PhD supervisor.

Dr. Simon Tsang (Peraton Labs)

Provided design feedback as a end-user.

A Word From Our Sponsors

This research was sponsored in part by the **United States Defense Advanced Research Projects Agency** (DARPA) under the **Quantum Benchmarking program, contract #HR001122C0066**.

Objectives and Overview

A Tale of Two Languages

A Tale of Two Languages

Perceval

Qiskit

TKET

Qiskit

Ket

Cirq

Silq

OpenQASM 3

Quipper

A Tale of Two Languages

Perceval

Qiskit

TKET

Qiskit

Ket

Cirq

Silq

OpenQASM 3

Quipper

A Tale of Two Languages

Perceval

Qiskit

TKET

Qiskit

Ket

Cirq

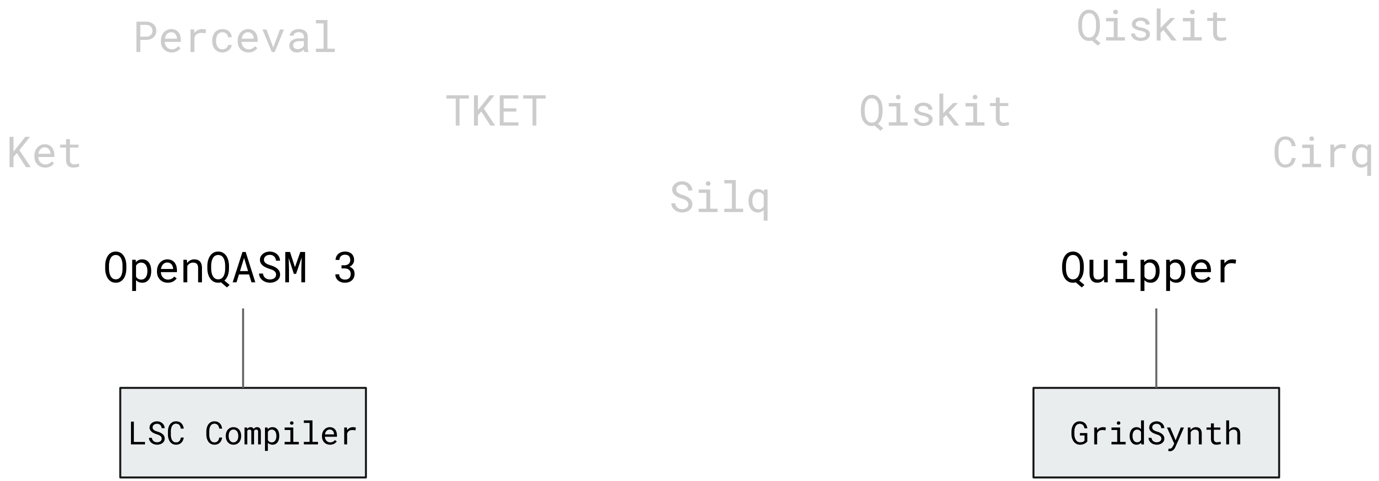
Silq

OpenQASM 3

Quipper

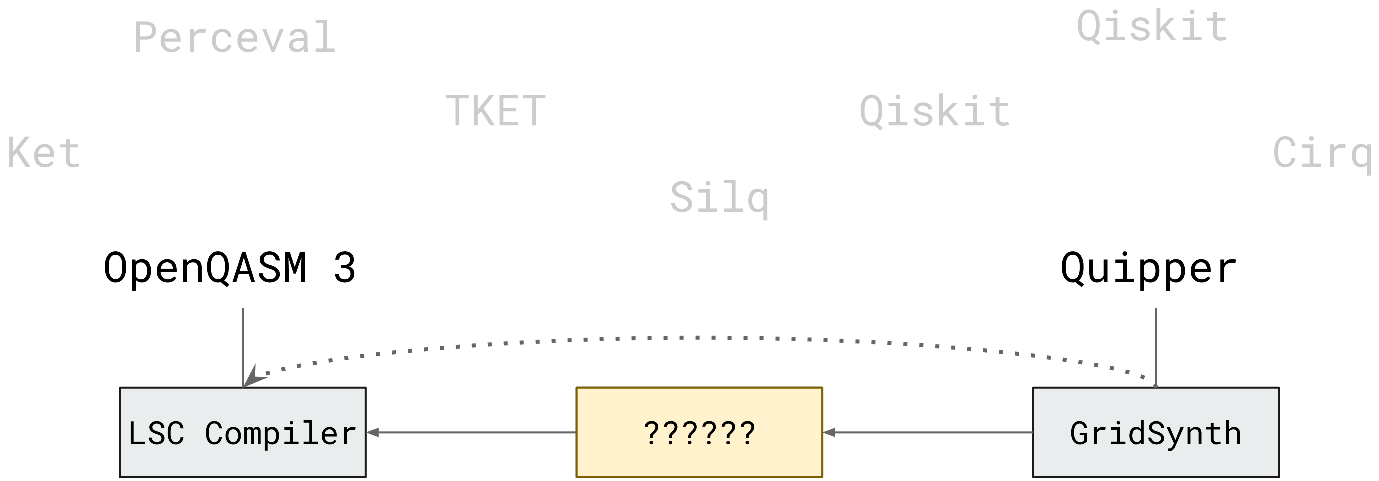
GridSynth

A Tale of Two Languages

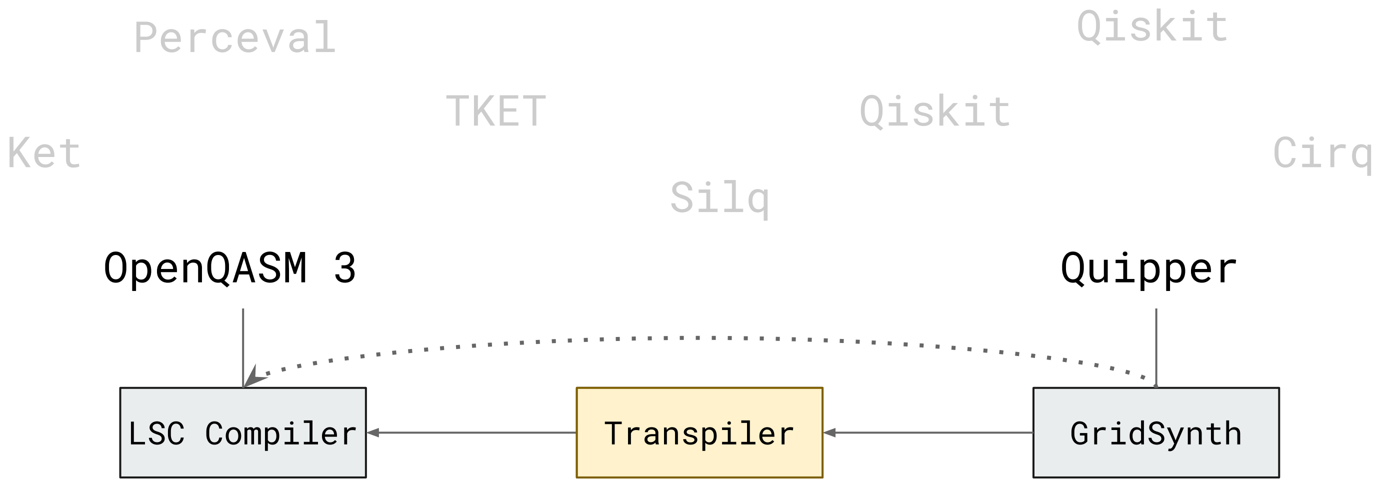


A Tale of Two Languages

A Tale of Two Languages



A Tale of Two Languages



Categorical Specification

How Should an Ideal Transpiler Behave?

The UNIX Design Philosophy and Pipelines

Implications of the UNIX philosophy:

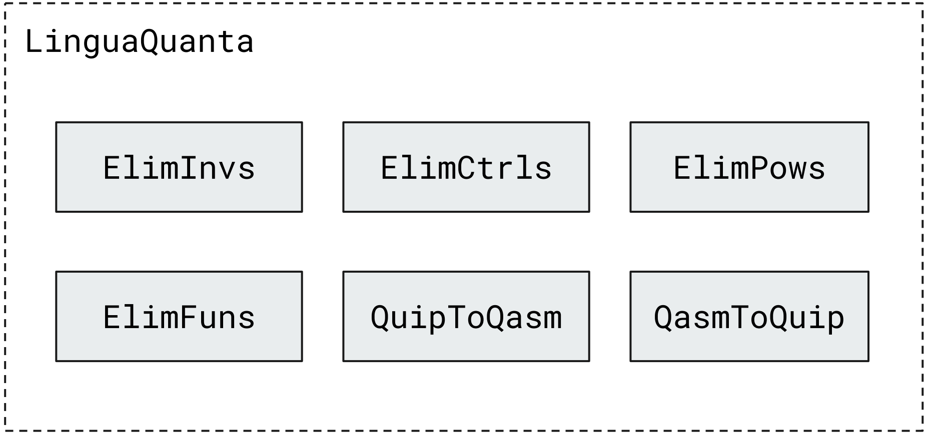
1.
2.
3.

LinguaQuanta

The UNIX Design Philosophy and Pipelines

Implications of the UNIX philosophy:

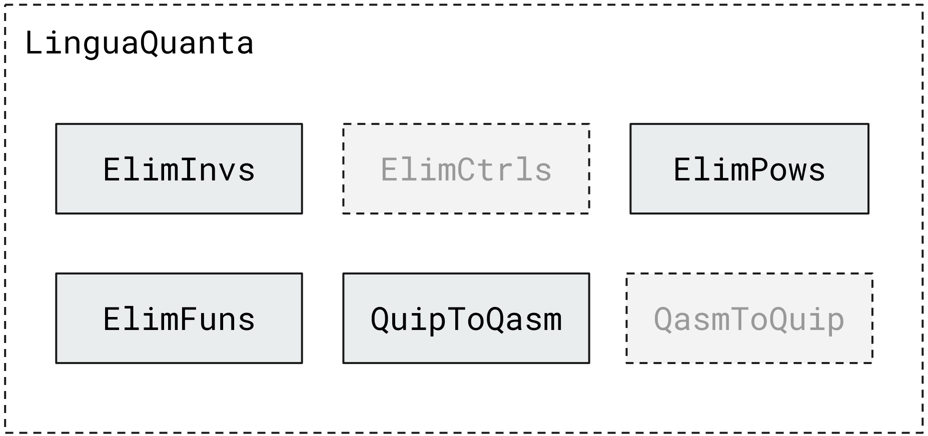
1. Write programs that do one thing and do it well.
2.
3.

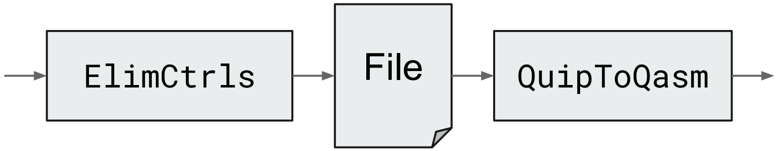


The UNIX Design Philosophy and Pipelines

Implications of the UNIX philosophy:

1. Write programs that do one thing and do it well.
2. Write programs that work well together.
3.

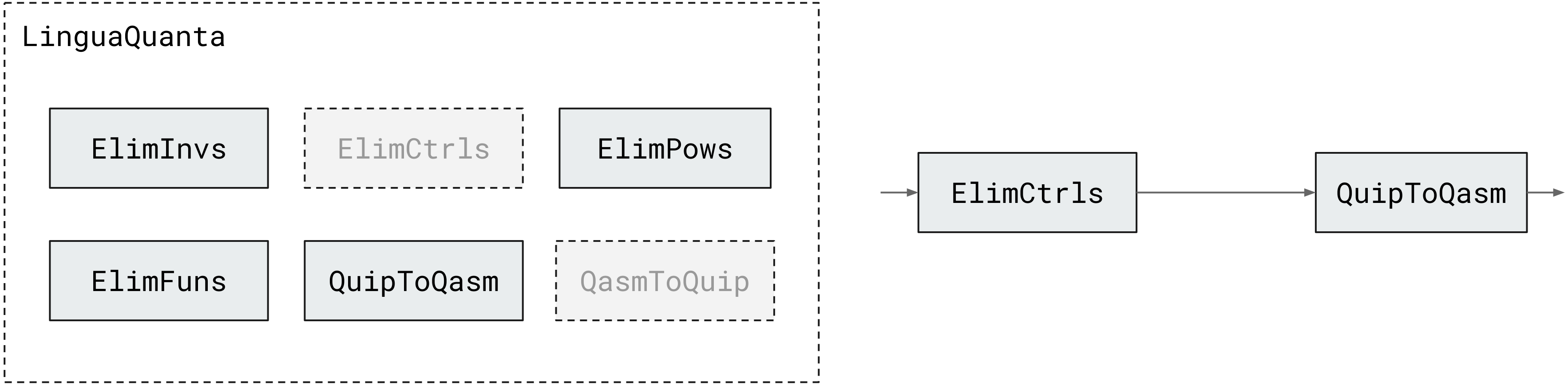




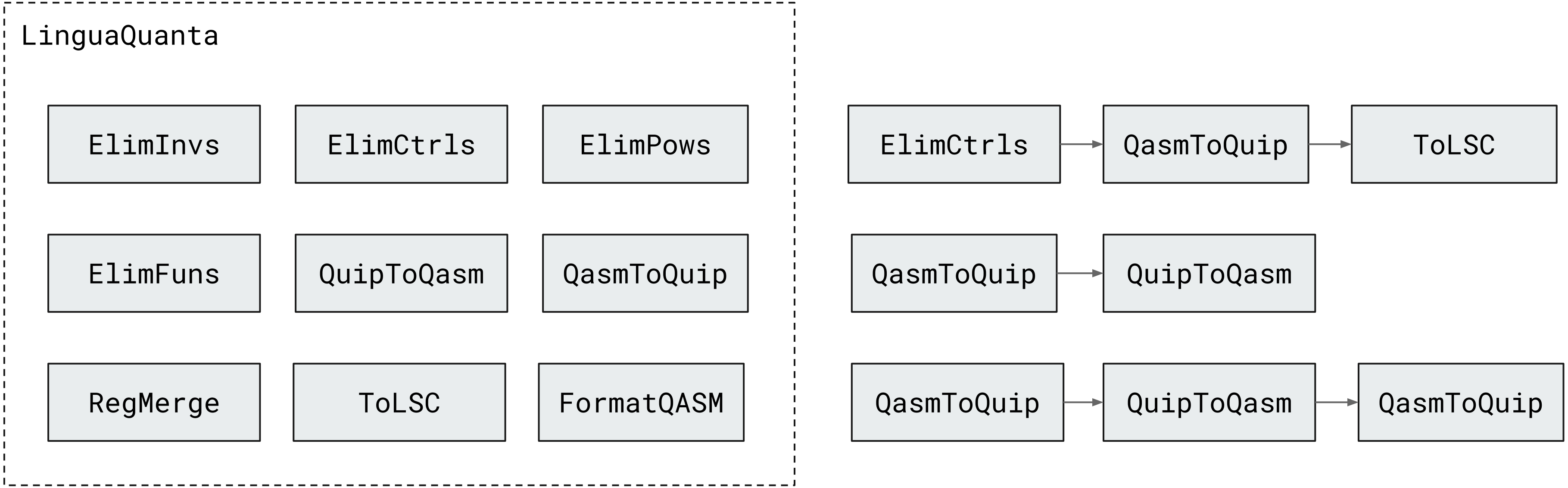
The UNIX Design Philosophy and Pipelines

Implications of the UNIX philosophy:

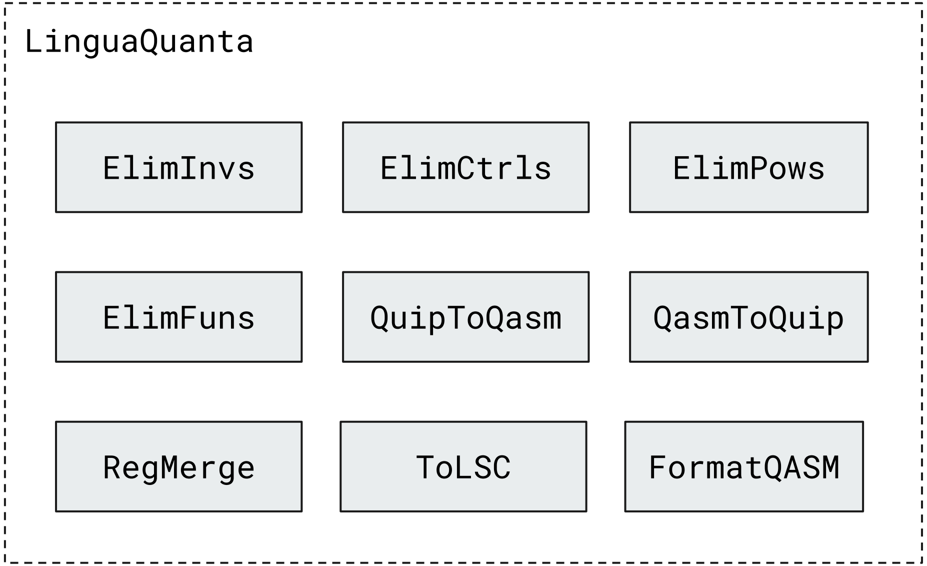
1. Write programs that do one thing and do it well.
2. Write programs that work well together.
3. Write programs that handle text streams.



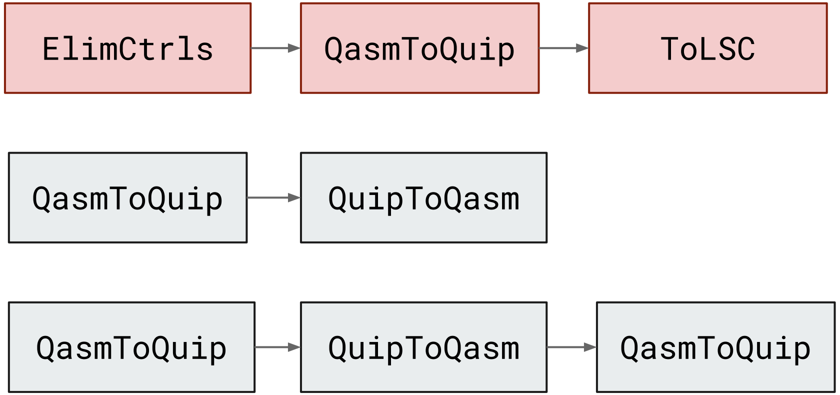
Problem: Many Potential Pipelines



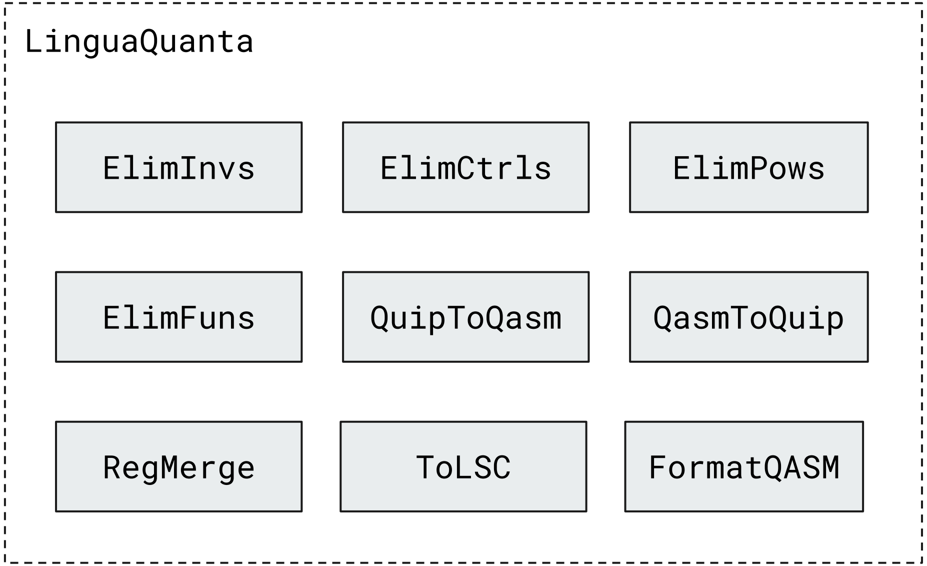
Problem: Many Potential Pipelines



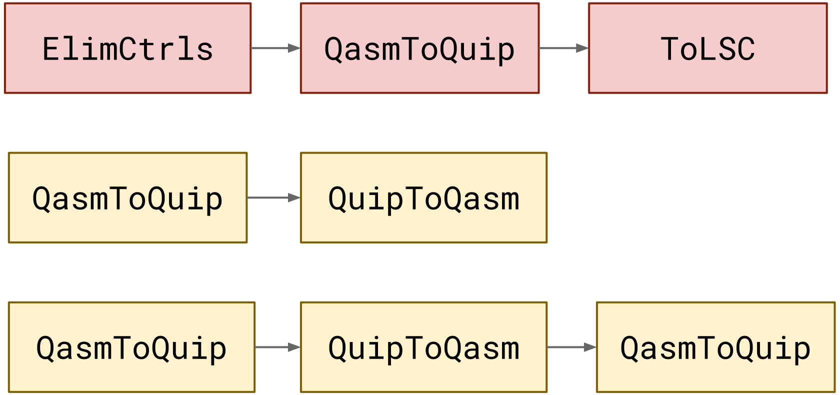
The output file types do not align with the input file types.



Problem: Many Potential Pipelines

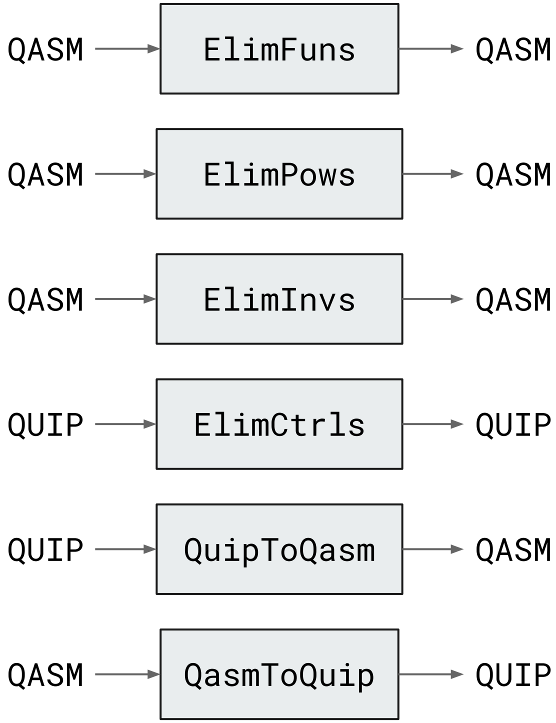


The output file types do not align with the input file types.

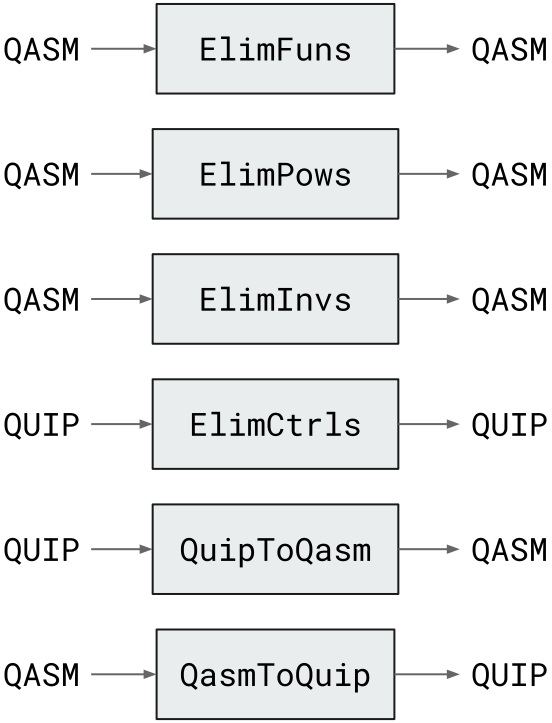


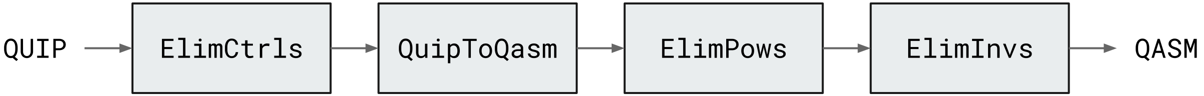
What should we expect from a round translation in LinguaQuanta?

Solution: Study the Semantics of Pipelines

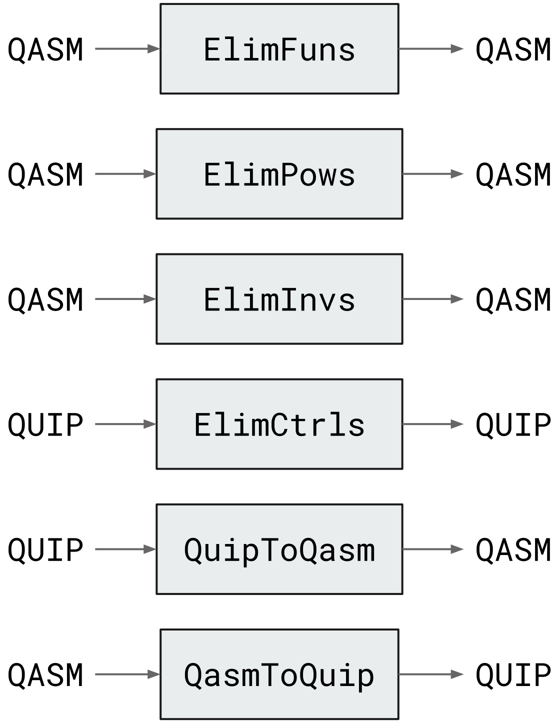


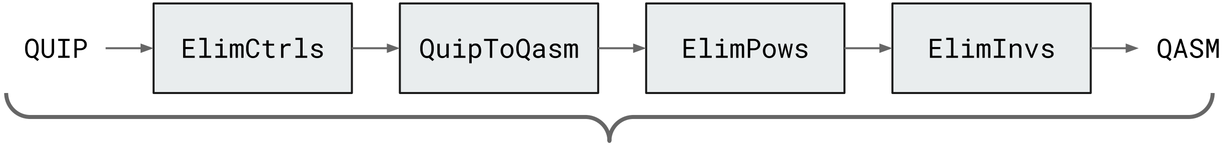
Solution: Study the Semantics of Pipelines





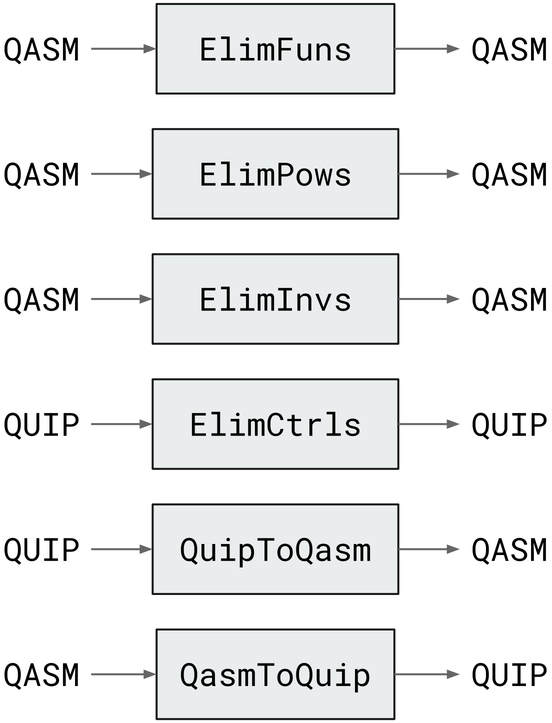
Solution: Study the Semantics of Pipelines





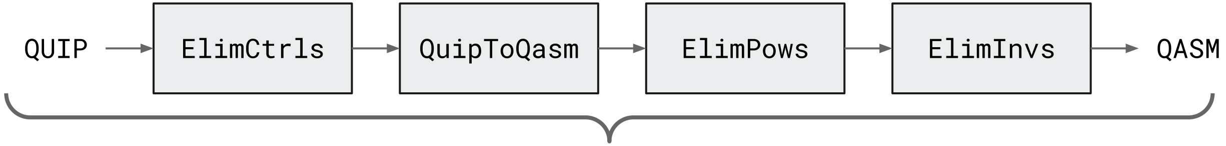
A portion of the OpenQASM 2.0 legacy support pipeline.

Solution: Study the Semantics of Pipelines



Now our pipelines are **type safe** in some very coarse grained sense. However, we would like to have a more fine-grained **notion of correctness**. Ideally, we would be able to **validate these specifications algorithmically**.

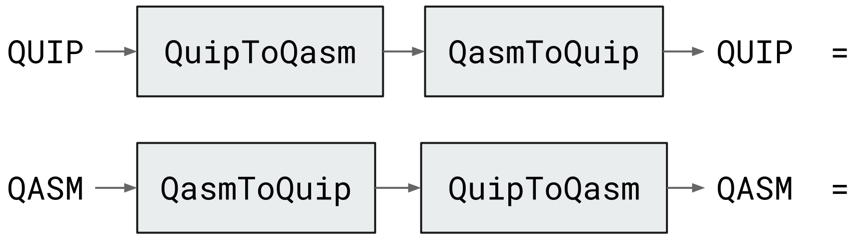
The rest of this section will refine these specifications to a satisfactory level, and then suggest a **direction for future work**.



A portion of the OpenQASM 2.0 legacy support pipeline.

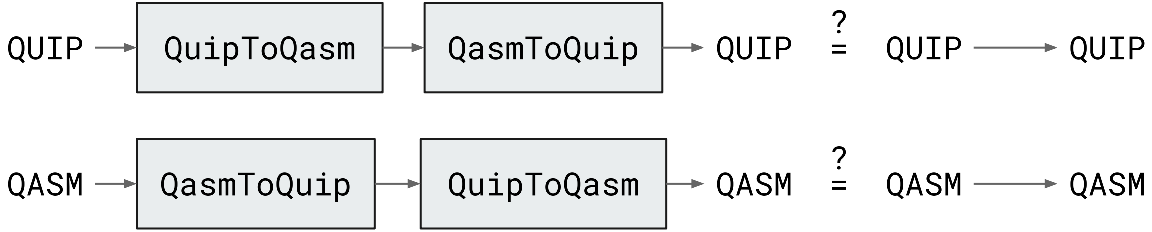
Step 1: Round Translations (Idempotents)

Q. What should happen to a round trip translation?



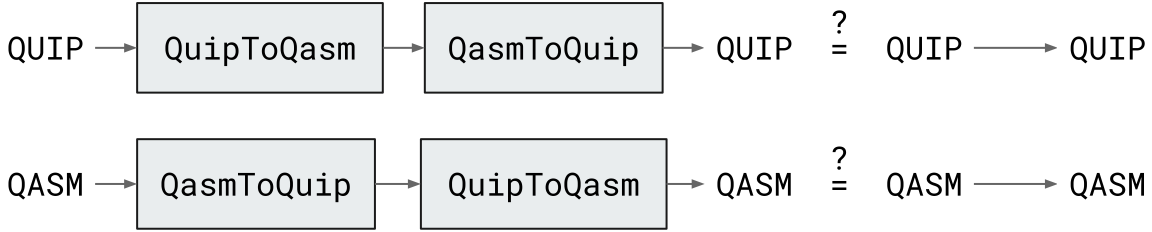
Step 1: Round Translations (Idempotents)

Q. What should happen to a round trip translation?

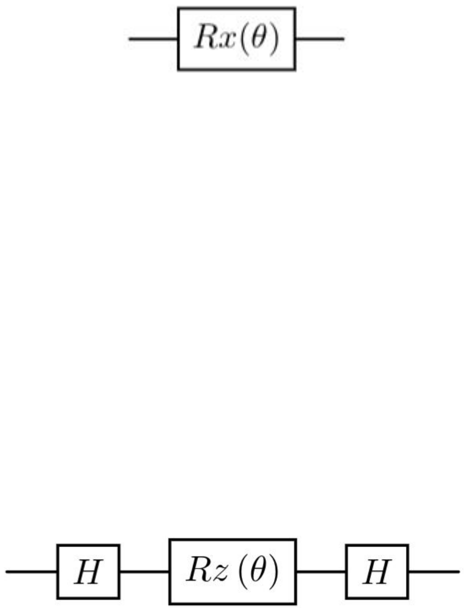


Step 1: Round Translations (Idempotents)

Q. What should happen to a round trip translation?

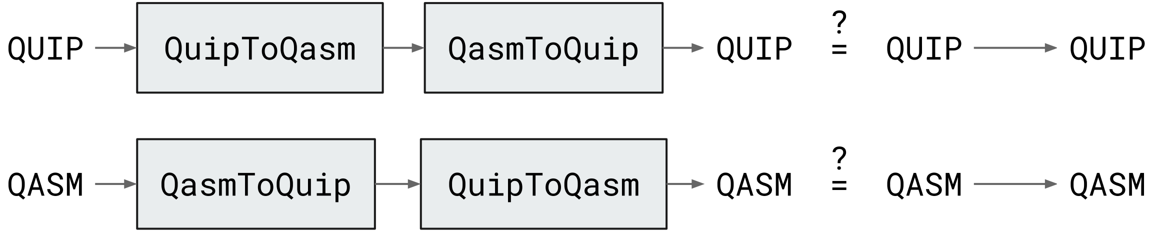


Counterexample to Proposal

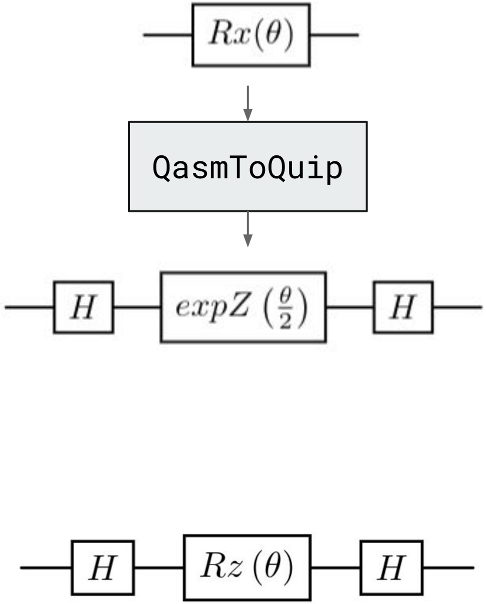


Step 1: Round Translations (Idempotents)

Q. What should happen to a round trip translation?

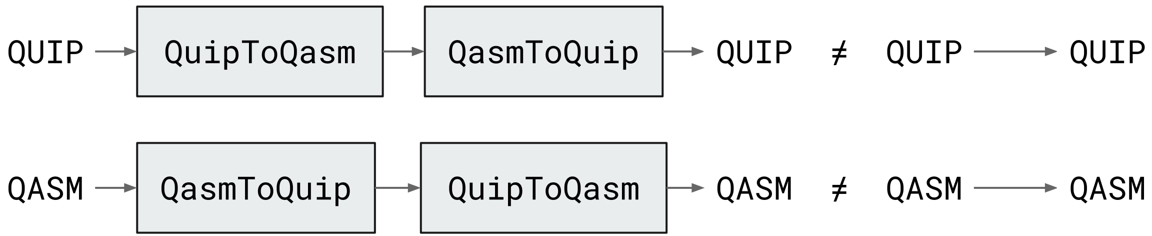


Counterexample to Proposal

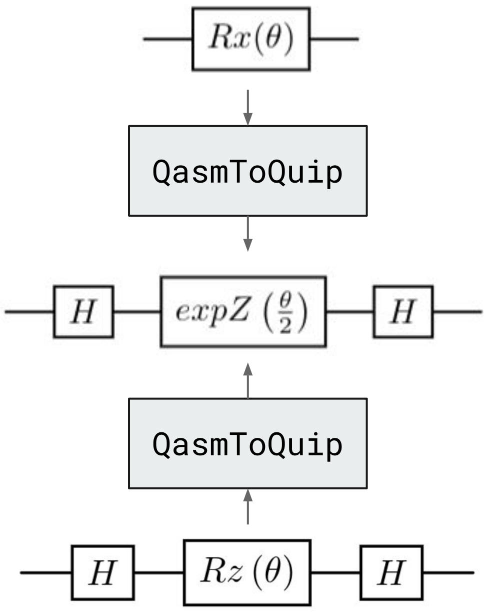


Step 1: Round Translations (Idempotents)

Q. What should happen to a round trip translation?

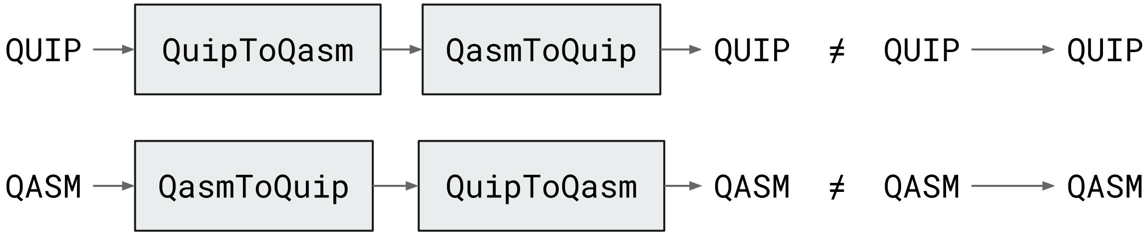


Counterexample to Proposal



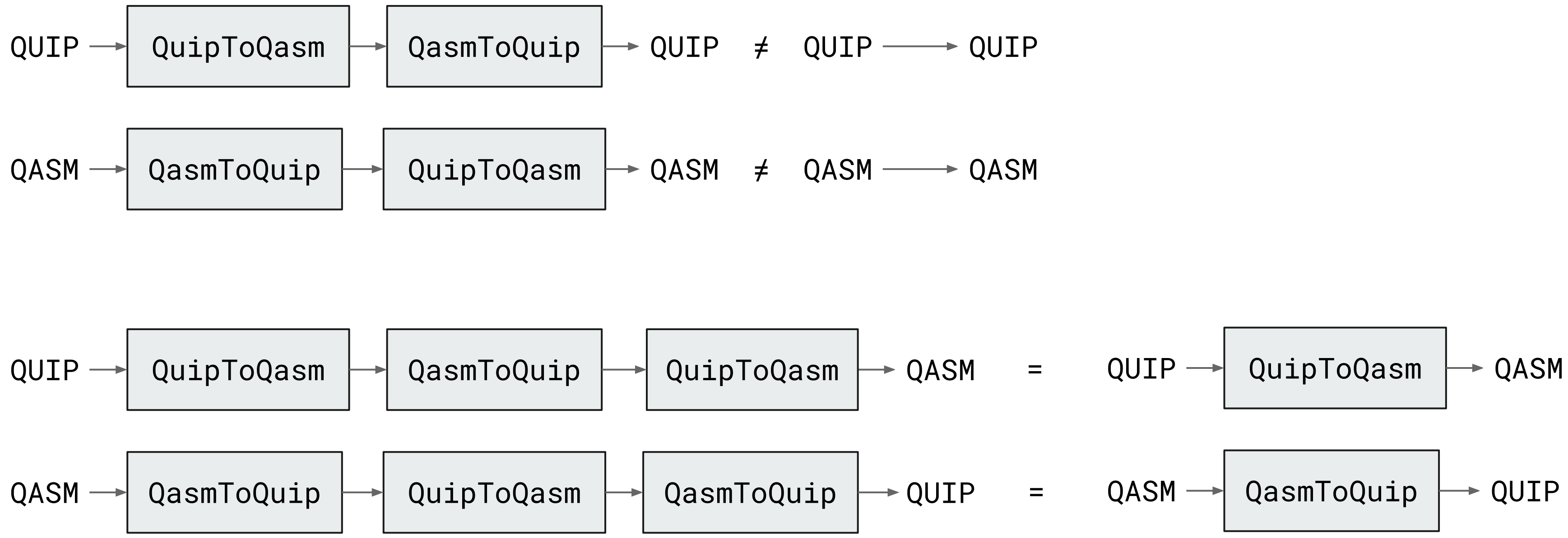
Step 1: Round Translations (Idempotents)

Q. What should happen to a round trip translation?



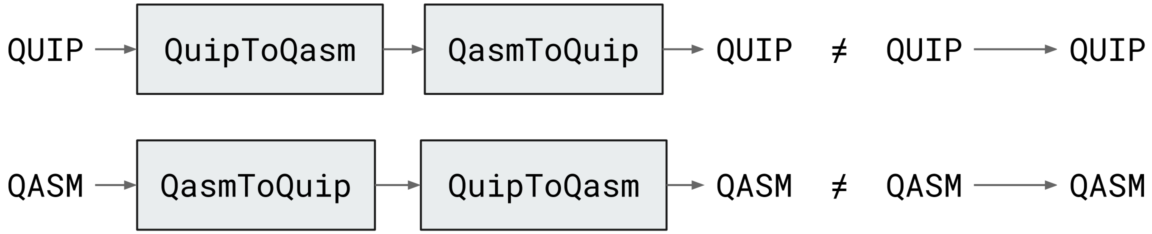
Step 1: Round Translations (Idempotents)

Q. What should happen to a round trip translation?



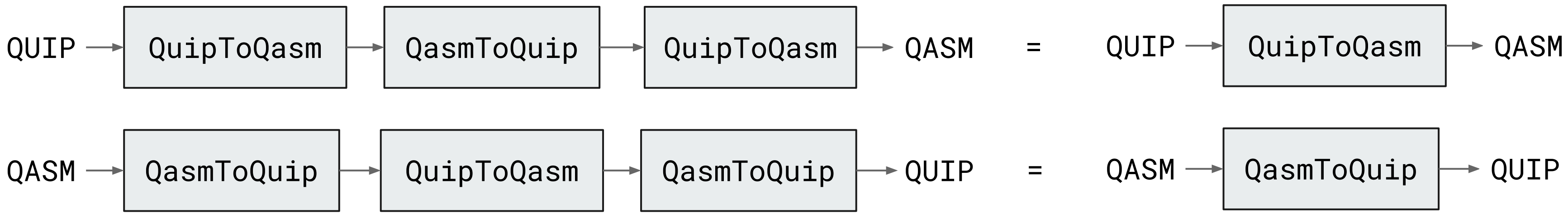
Step 1: Round Translations (Idempotents)

Q. What should happen to a round trip translation?



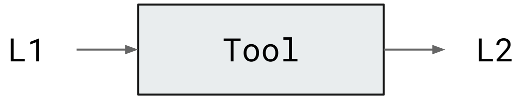
This set of requirements entail the **idempotence** of **round translations**.

We argue that these are reasonable properties for a transpiler to satisfy. They capture requirements such as round translates are **stable**, and **will not inflate** file size.



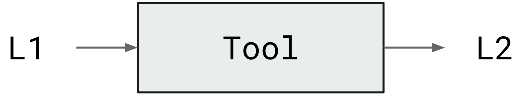
Step 2: A Closer Look at Translations

Q: Is there more structure to a transpilation stage?

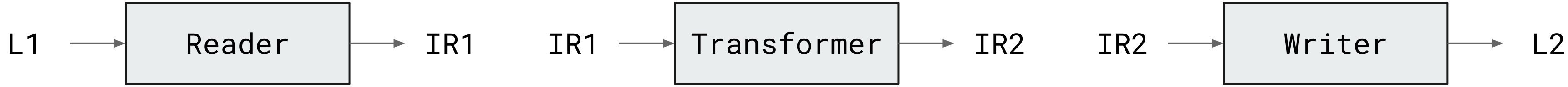


Step 2: A Closer Look at Translations

Q: Is there more structure to a transpilation stage?

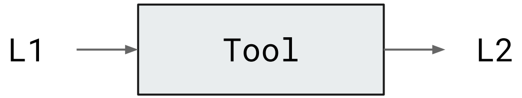


We know from compiler design that **operating directly on source text is a bad design pattern**… at the very least, we should have a "**reader**" (lexer + parser) and a "**writer**" (code generation phase).

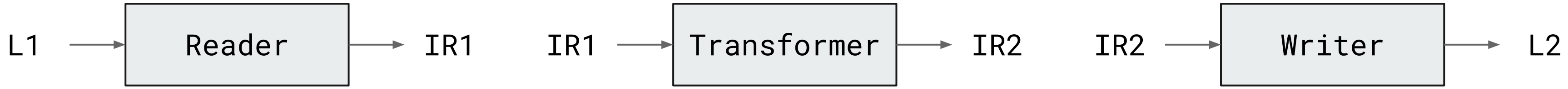


Step 2: A Closer Look at Translations

Q: Is there more structure to a transpilation stage?



We know from compiler design that **operating directly on source text is a bad design pattern**… at the very least, we should have a "**reader**" (lexer + parser) and a "**writer**" (code generation phase).



This yields the following definition:

Step 3: Read-What-You-Write (Retracts)

Q: How should readers and writers interact?

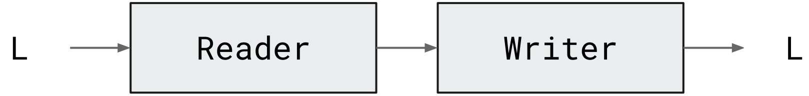
Step 3: Read-What-You-Write (Retracts)

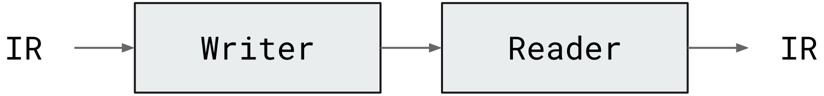
Q: How should readers and writers interact?

It is possible that **many syntactic structures** map to the same internal representation (IR). For example, pairs of inverse modifiers cancel out in OpenQASM 3. However, each IR statement can have a **canonical representation**.

Step 3: Read-What-You-Write (Retracts)

Q: How should readers and writers interact?





It is possible that **many syntactic structures** map to the same internal representation (IR). For example, pairs of inverse modifiers cancel out in OpenQASM 3. However, each IR statement can have a **canonical representation**.

Categorically, this says that the reader should provide a **retraction** (left-inverse) for the writer.

An Example Retraction With OpenQASM

```
qubit q;
inv @ s q;
```

```
qubit q;
inv @ inv @ inv @ s q;
```

```
qubit q;
inv @ inv @ inv
    @ inv @ inv @ s q;
```

An Example Retraction With OpenQASM

```
qubit q;
inv @ s q;
```

```
qubit q;
inv @ inv @ inv @ s q;
```

```
qubit q;
inv @ inv @ inv
    @ inv @ inv @ s q;
```

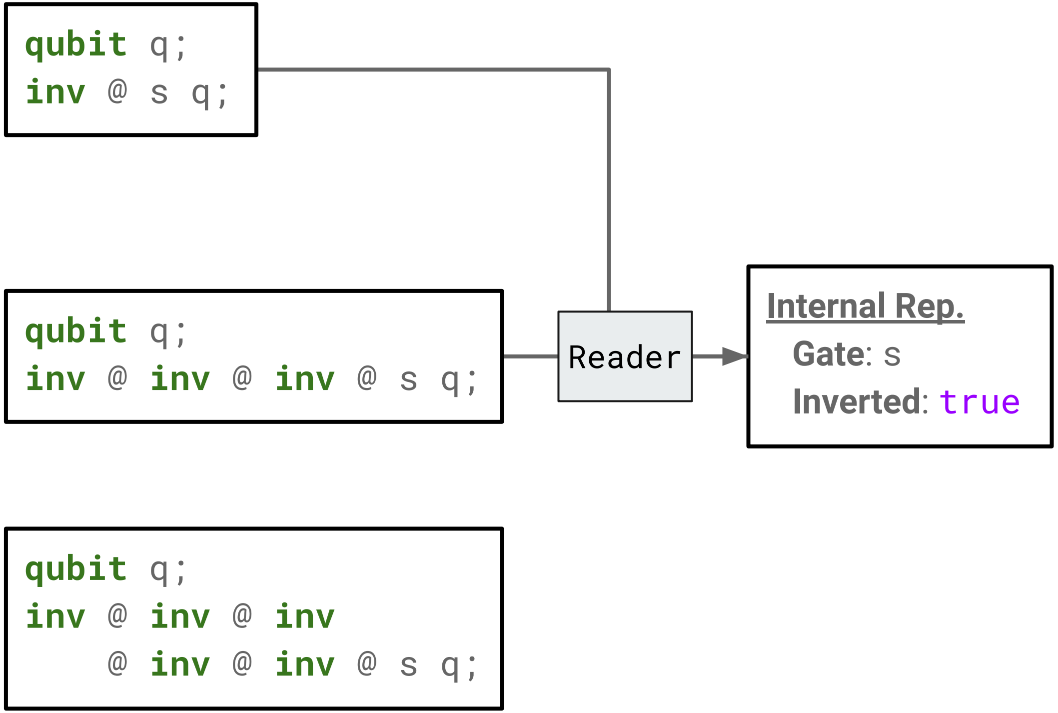
Reader →

Internal Rep.

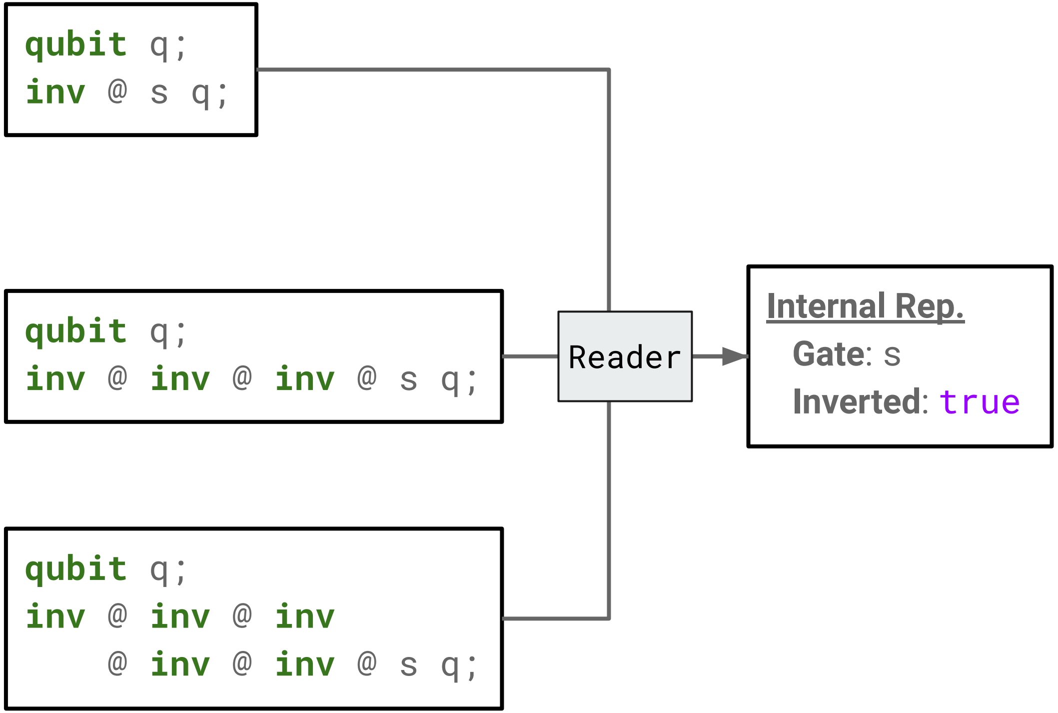
Gate: s

Inverted: true

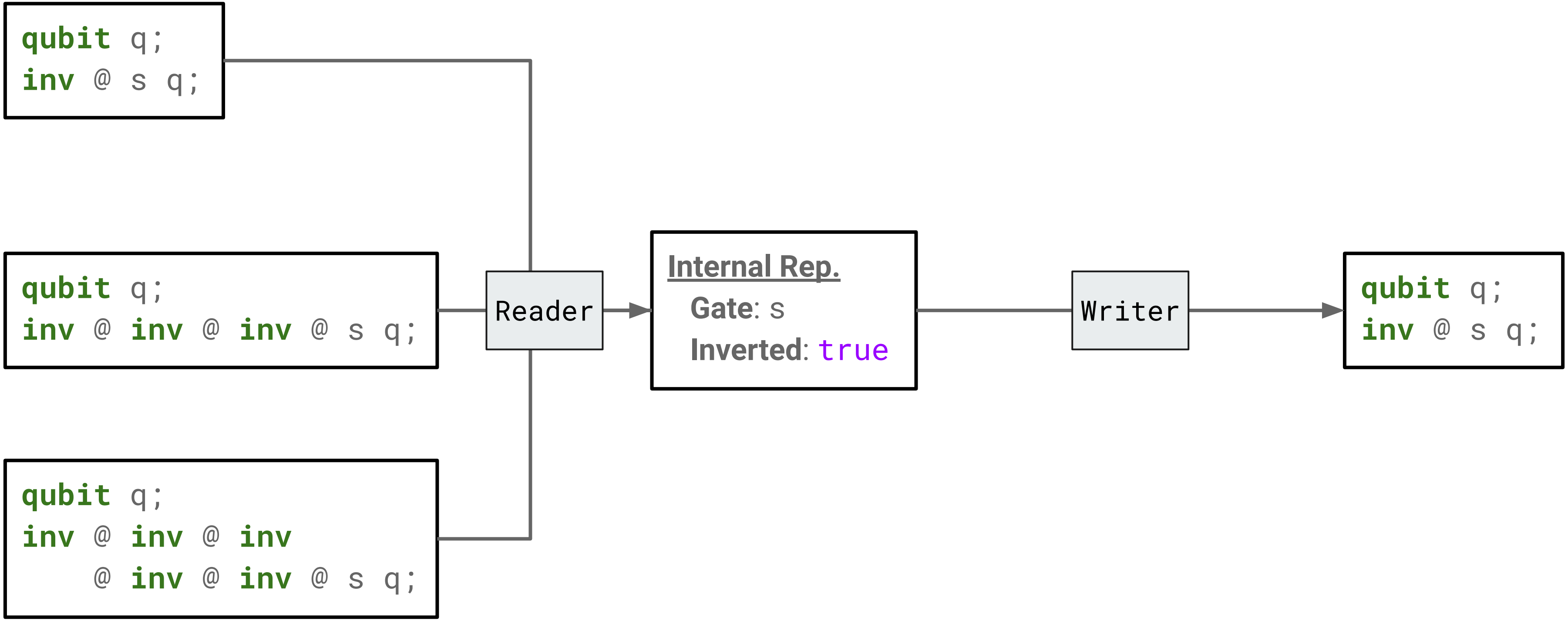
An Example Retraction With OpenQASM



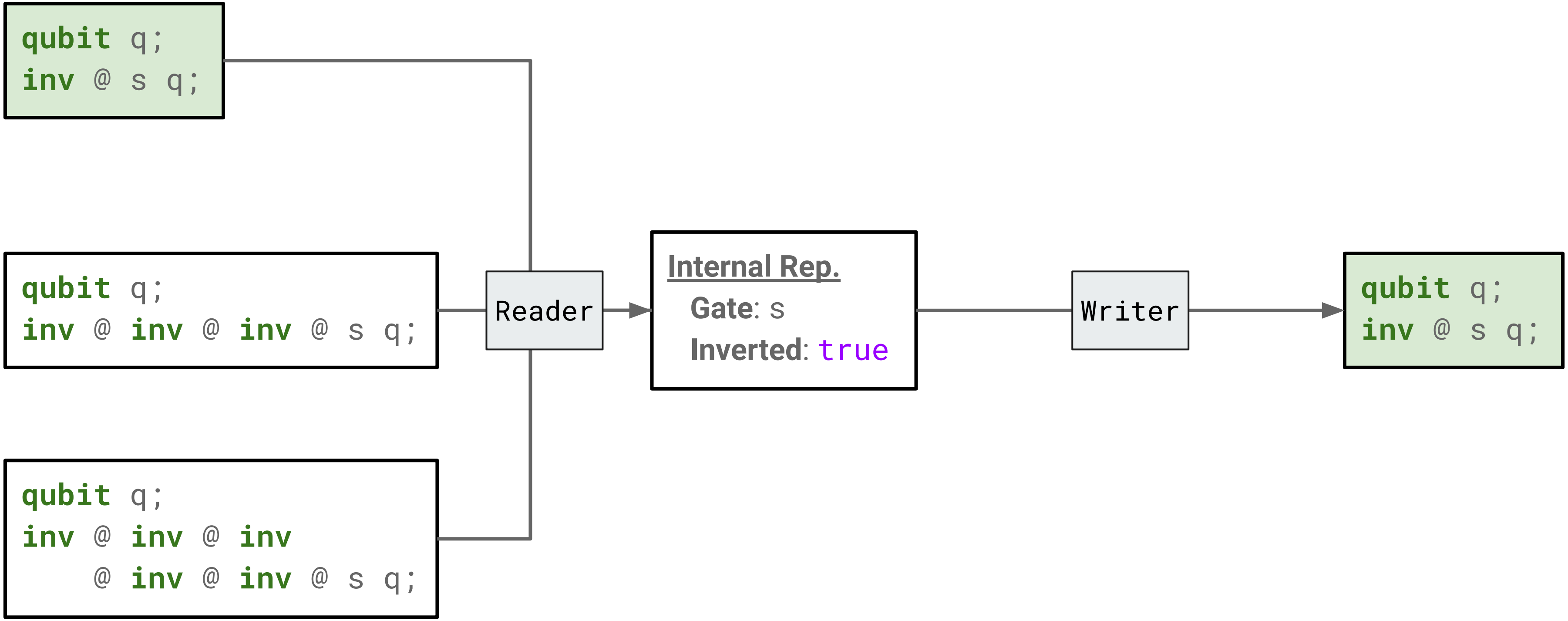
An Example Retraction With OpenQASM



An Example Retraction With OpenQASM



An Example Retraction With OpenQASM



Some Decompositions

Rewriting for a Quantum Transpiler?

OpenQASM and Quipper Lack Feature Parity

Notation	Matrix	QUIP	QASM3	QASM2
$\oplus$	$\begin{bmatrix}0&1\\1&0\end{bmatrix}$	✓	✓	✓
Y	$\begin{bmatrix}0&-i\\i&0\end{bmatrix}$	✓	✓	✓
Z	$\begin{bmatrix}1&0\\0&-1\end{bmatrix}$	✓	✓	✓
H	$\frac{1}{\sqrt{2}}\begin{bmatrix}1&1\\1&-1\end{bmatrix}$	✓	✓	✓
S	$\begin{bmatrix}1&0\\0&i\end{bmatrix}$	✓	✓	✓
$S^\dagger$	$\begin{bmatrix}1&0\\0&-i\end{bmatrix}$	✓	✓	✓
T	$\begin{bmatrix}1&0\\0&\omega\end{bmatrix}$	✓	✓	✓
$T^\dagger$	$\begin{bmatrix}1&0\\0&\omega^7\end{bmatrix}$	✓	✓	✓
sx	$\frac{1}{2}\begin{bmatrix}1+i&1-i\\1-i&1+i\end{bmatrix}$	✓	✓	×
iX	$\begin{bmatrix}0&i\\i&0\end{bmatrix}$	✓	×	×
ω	$\begin{bmatrix}\omega&0\\0&\omega\end{bmatrix}$	✓	×	×
E	$\frac{1}{2}\begin{bmatrix}-1+i&1+i\\-1+i&-1-i\end{bmatrix}$	✓	×	×

Notation	Matrix	QUIP	QASM3	QASM2
	$I \oplus X$	✓	✓	✓
Y	$I \oplus Y$	✓	✓	✓
	$I \oplus Z$	✓	✓	✓
H	$I \oplus H$	✓	✓	✓
	$\begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix}$	✓	✓	×
W	$\begin{bmatrix}1&0&0&0\\0&\frac{1}{\sqrt{2}}&\frac{1}{\sqrt{2}}&0\\0&\frac{1}{\sqrt{2}}&-\frac{1}{\sqrt{2}}&0\\0&0&0&1\end{bmatrix}$	✓	×	×
	$I \oplus I \oplus X$	✓	✓	✓
	$I \oplus SWAP$	✓	✓	×

Notation	Matrix	QUIP	QASM3	QASM2
$expZ(\theta)$	$e^{-iZ\theta}$	✓	×	×
$R_x(\theta)$	$e^{-iX(\theta/2)}$	×	✓	✓
$R_y(\theta)$	$e^{-iY(\theta/2)}$	×	✓	✓
$R_z(\theta)$	$e^{-iZ(\theta/2)}$	×	✓	✓
$P(\theta)$	$\begin{bmatrix}1&0\\0&e^{i\theta}\end{bmatrix}$	✓	✓	×
$U(\theta,\phi,\lambda)$	See Appx. A.	×	✓	✓
$U3(\theta,\phi,\lambda)$	See Appx. A.	×	✓	✓
$U2(\phi,\lambda)$	See Appx. A.	×	✓	✓
$U1(\lambda)$	See Appx. A.	×	✓	✓

Notation	Matrix	QUIP	QASM3	QASM2
$R_x(\theta)$	$I \oplus R_x(\theta)$	×	✓	×
$R_y(\theta)$	$I \oplus R_y(\theta)$	×	✓	×
$R_z(\theta)$	$I \oplus R_z(\theta)$	×	✓	✓
$P(\theta)$	$I \oplus P(\theta)$	×	✓	×
$CU(\gamma,\theta,\rho,\lambda)$	$e^{i\gamma} \cdot U(\theta,\rho,\lambda)$	×	✓	×

OpenQASM and Quipper Lack Feature Parity

Notation	Matrix	QUIP	QASM3	QASM2
$\oplus$	$\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$	✓	✓	✓
Y	$\begin{bmatrix}0 & -i\\i & 0\end{bmatrix}$	✓	✓	✓
Z	$\begin{bmatrix}1 & 0\\0 & -1\end{bmatrix}$	✓	✓	✓
H	$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}$	✓	✓	✓
S	$\begin{bmatrix}1 & 0\\0 & i\end{bmatrix}$	✓	✓	✓
$S^\dagger$	$\begin{bmatrix}1 & 0\\0 & -i\end{bmatrix}$	✓	✓	✓
T	$\begin{bmatrix}1 & 0\\0 & \omega\end{bmatrix}$	✓	✓	✓
$T^\dagger$	$\begin{bmatrix}1 & 0\\0 & \omega^7\end{bmatrix}$	✓	✓	✓
sx	$\frac{1}{2}\begin{bmatrix}1+i & 1-i\\1-i & 1+i\end{bmatrix}$	✓	✓	×
iX	$\begin{bmatrix}0 & i\\i & 0\end{bmatrix}$	✓	×	×
ω	$\begin{bmatrix}\omega & 0\\0 & \omega\end{bmatrix}$	✓	×	×
E	$\frac{1}{2}\begin{bmatrix}-1+i & 1+i\\-1+i & -1-i\end{bmatrix}$	✓	×	×

Notation	Matrix	QUIP	QASM3	QASM2
	$I \oplus X$	✓	✓	✓
Y	$I \oplus Y$	✓	✓	✓
	$I \oplus Z$	✓	✓	✓
H	$I \oplus H$	✓	✓	✓
	$\begin{bmatrix}1 & 0 & 0 & 0\\0 & 0 & 1 & 0\\0 & 1 & 0 & 0\\0 & 0 & 0 & 1\end{bmatrix}$	✓	✓	×
W	$\begin{bmatrix}1 & 0 & 0 & 0\\0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0\\0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0\\0 & 0 & 0 & 1\end{bmatrix}$	✓	×	×
	$I \oplus I \oplus X$	✓	✓	✓
	$I \oplus SWAP$	✓	✓	×

Notation	Matrix	QUIP	QASM3	QASM2
$expZ(\theta)$	$e^{-iZ\theta}$	✓	×	×
$R_x(\theta)$	$e^{-iX(\theta/2)}$	×	✓	✓
$R_y(\theta)$	$e^{-iY(\theta/2)}$	×	✓	✓
$R_z(\theta)$	$e^{-iZ(\theta/2)}$	×	✓	✓
$P(\theta)$	$\begin{bmatrix}1 & 0\\0 & e^{i\theta}\end{bmatrix}$	✓	✓	×
$U(\theta, \phi, \lambda)$	See Appx. A.	×	✓	✓
$U3(\theta, \phi, \lambda)$	See Appx. A.	×	✓	✓
$U2(\phi, \lambda)$	See Appx. A.	×	✓	✓
$U1(\lambda)$	See Appx. A.	×	✓	✓

Notation	Matrix	QUIP	QASM3	QASM2
$R_x(\theta)$	$I \oplus R_x(\theta)$	×	✓	×
$R_y(\theta)$	$I \oplus R_y(\theta)$	×	✓	×
$R_z(\theta)$	$I \oplus R_z(\theta)$	×	✓	✓
$P(\theta)$	$I \oplus P(\theta)$	×	✓	×
$CU(\gamma, \theta, \rho, \lambda)$	$e^{i\gamma} \cdot U(\theta, \rho, \lambda)$	×	✓	×

OpenQASM and Quipper Lack Feature Parity

Notation	Matrix	QUIP	QASM3	QASM2
$\oplus$	$\begin{bmatrix}0&1\\1&0\end{bmatrix}$	✓	✓	✓
Y	$\begin{bmatrix}0&-i\\i&0\end{bmatrix}$	✓	✓	✓
Z	$\begin{bmatrix}1&0\\0&-1\end{bmatrix}$	✓	✓	✓
H	$\frac{1}{\sqrt{2}}\begin{bmatrix}1&1\\1&-1\end{bmatrix}$	✓	✓	✓
S	$\begin{bmatrix}1&0\\0&i\end{bmatrix}$	✓	✓	✓
$S^\dagger$	$\begin{bmatrix}1&0\\0&-i\end{bmatrix}$	✓	✓	✓
T	$\begin{bmatrix}1&0\\0&\omega\end{bmatrix}$	✓	✓	✓
$T^\dagger$	$\begin{bmatrix}1&0\\0&\omega^7\end{bmatrix}$	✓	✓	✓
sx	$\frac{1}{2}\begin{bmatrix}1+i&1-i\\1-i&1+i\end{bmatrix}$	✓	✓	×
iX	$\begin{bmatrix}0&i\\i&0\end{bmatrix}$	✓	×	×
ω	$\begin{bmatrix}\omega&0\\0&\omega\end{bmatrix}$	✓	×	×
E	$\frac{1}{2}\begin{bmatrix}-1+i&1+i\\-1+i&-1-i\end{bmatrix}$	✓	×	×

Notation	Matrix	QUIP	QASM3	QASM2
controlled $\oplus$	$I \oplus X$	✓	✓	✓
controlled Y	$I \oplus Y$	✓	✓	✓
controlled Z	$I \oplus Z$	✓	✓	✓
controlled H	$I \oplus H$	✓	✓	✓
swap	$\begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix}$	✓	✓	×
W	$\begin{bmatrix}1&0&0&0\\0&\frac{1}{\sqrt{2}}&\frac{1}{\sqrt{2}}&0\\0&\frac{1}{\sqrt{2}}&-\frac{1}{\sqrt{2}}&0\\0&0&0&1\end{bmatrix}$	✓	×	×
doubly controlled $\oplus$	$I \oplus I \oplus X$	✓	✓	✓
controlled swap	$I \oplus SWAP$	✓	✓	×

Notation	Matrix	QUIP	QASM3	QASM2
$expZ(\theta)$	$e^{-iZ\theta}$	✓	×	×
$R_x(\theta)$	$e^{-iX(\theta/2)}$	×	✓	✓
$R_y(\theta)$	$e^{-iY(\theta/2)}$	×	✓	✓
$R_z(\theta)$	$e^{-iZ(\theta/2)}$	×	✓	✓
$P(\theta)$	$\begin{bmatrix}1&0\\0&e^{i\theta}\end{bmatrix}$	✓	✓	×
$U(\theta,\phi,\lambda)$	See Appx. A.	×	✓	✓
$U3(\theta,\phi,\lambda)$	See Appx. A.	×	✓	✓
$U2(\phi,\lambda)$	See Appx. A.	×	✓	✓
$U1(\lambda)$	See Appx. A.	×	✓	✓

Notation	Matrix	QUIP	QASM3	QASM2
controlled $R_x(\theta)$	$I \oplus R_x(\theta)$	×	✓	×
controlled $R_y(\theta)$	$I \oplus R_y(\theta)$	×	✓	×
controlled $R_z(\theta)$	$I \oplus R_z(\theta)$	×	✓	✓
controlled $P(\theta)$	$I \oplus P(\theta)$	×	✓	×
$CU(\gamma,\theta,\rho,\lambda)$	$e^{i\gamma}\cdot U(\theta,\rho,\lambda)$	×	✓	×

Specific Decompositions for Specific Gates

Named gates without parameters typically require **one-off translations**. A part of this project was to bring together a list of these translations, in a "**Rewriting Compendium**", so to speak.

Specific Decompositions for Specific Gates

Named gates without parameters typically require **one-off translations**. A part of this project was to bring together a list of these translations, in a "**Rewriting Compendium**", so to speak.

Some statistics about this compendium:

1. **Number of References**: 9
2. **Number of Rewrite Rules**: 51
3. **Number of Non-Elementary Relations**: 41

This compendium is nowhere near comprehensive, but we hope for it to be a **start for compiler design**.

Specific Decompositions for Specific Gates

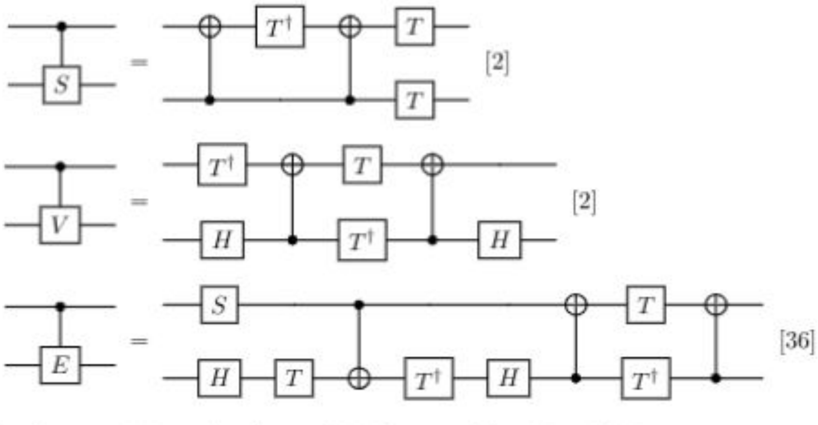
Named gates without parameters typically require **one-off translations**. A part of this project was to bring together a list of these translations, in a “**Rewriting Compendium**”, so to speak.

Some statistics about this compendium:

1. **Number of References**: 9
2. **Number of Rewrite Rules**: 51
3. **Number of Non-Elementary Relations**: 41

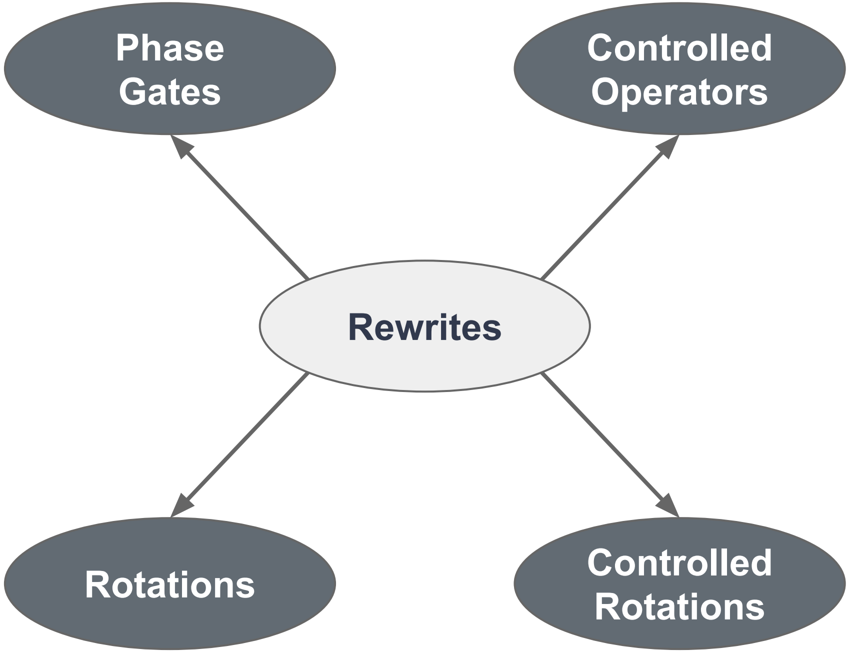
This compendium is nowhere near comprehensive, but we hope for it to be a **start for compiler design**.

Some Example Rewrite Rules:

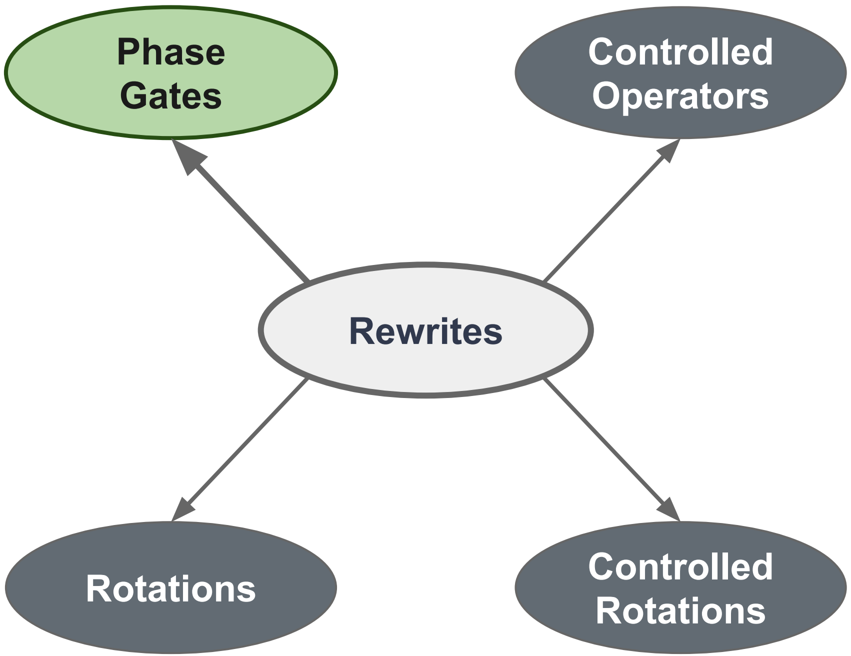


[2] M. Amy, D. Maslov, M. Mosca, M. Roetteler 2013.
[36] https://www.mathstat.dal.ca/~selinger/quipper/doc/

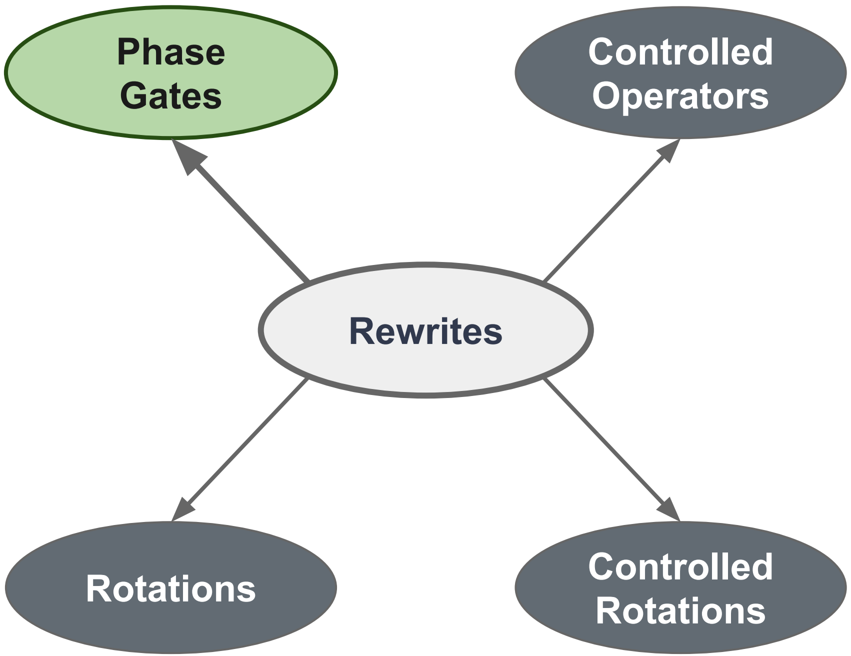
Wider Classes of Gate Decompositions



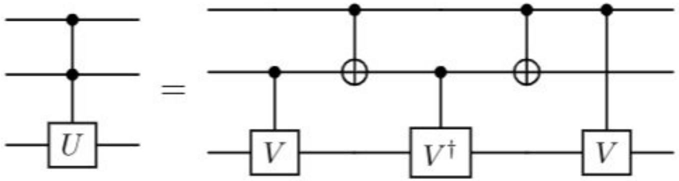
Wider Classes of Gate Decompositions



Wider Classes of Gate Decompositions

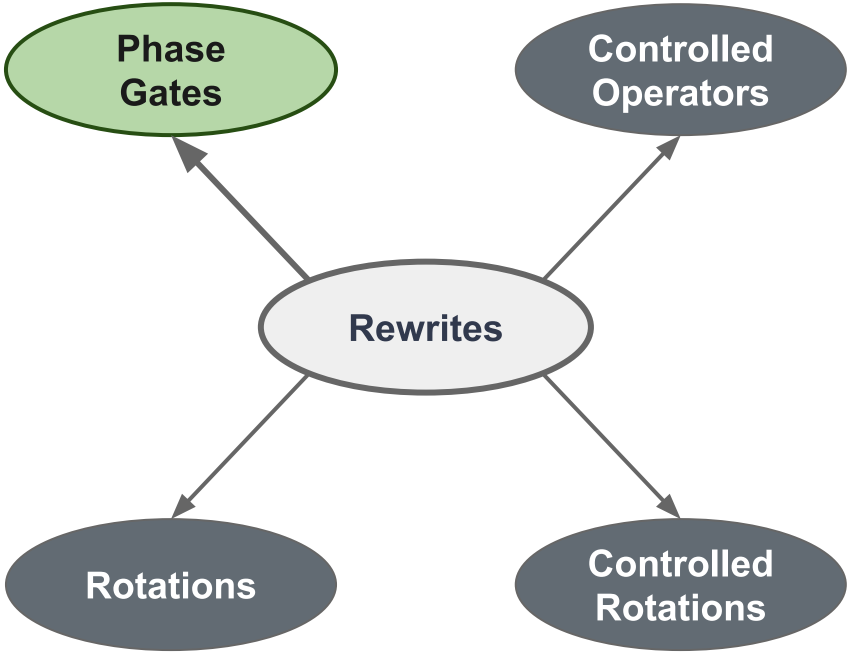


Assume that V is a unitary and $U = V^2$.

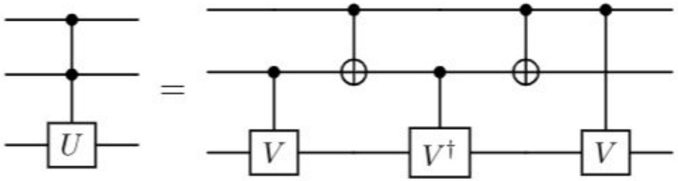


[4] A. Barenco et al., 1995.

Wider Classes of Gate Decompositions

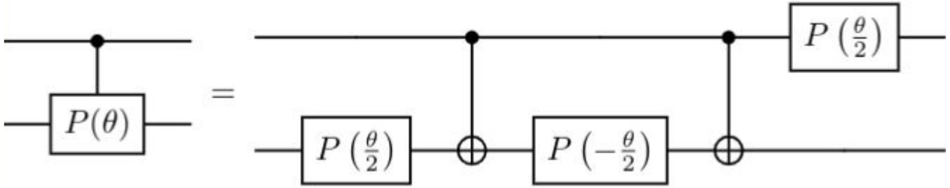


Assume that V is a unitary and U = V^2.

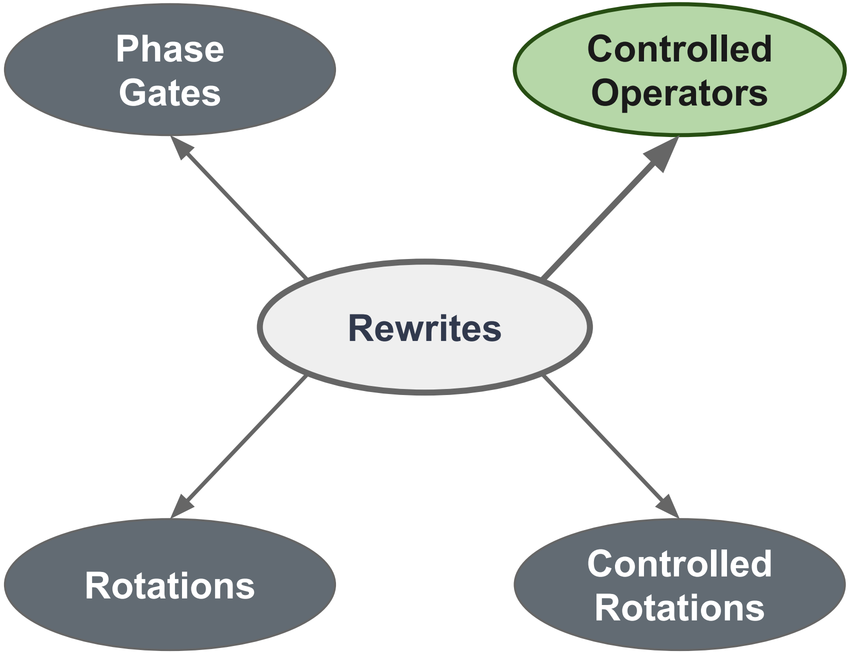


[4] A. Barenco et al., 1995.

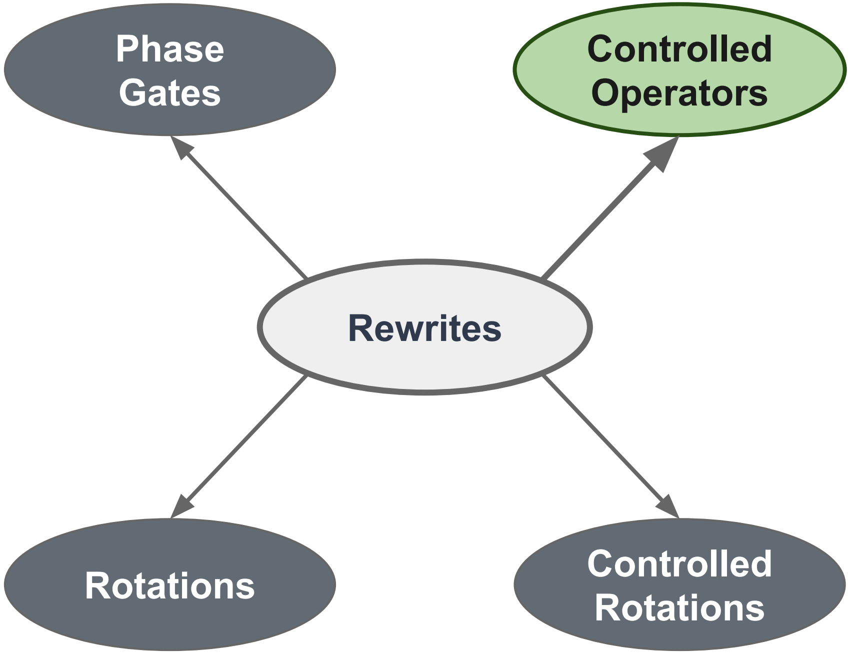
The P(θ) gate in OpenQASM is a controlled phase.

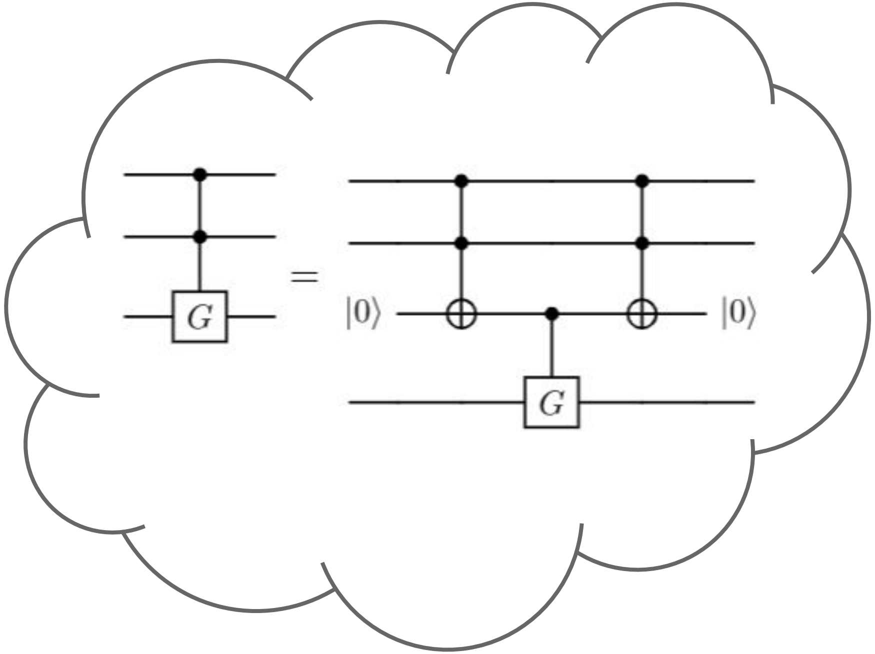


Wider Classes of Gate Decompositions

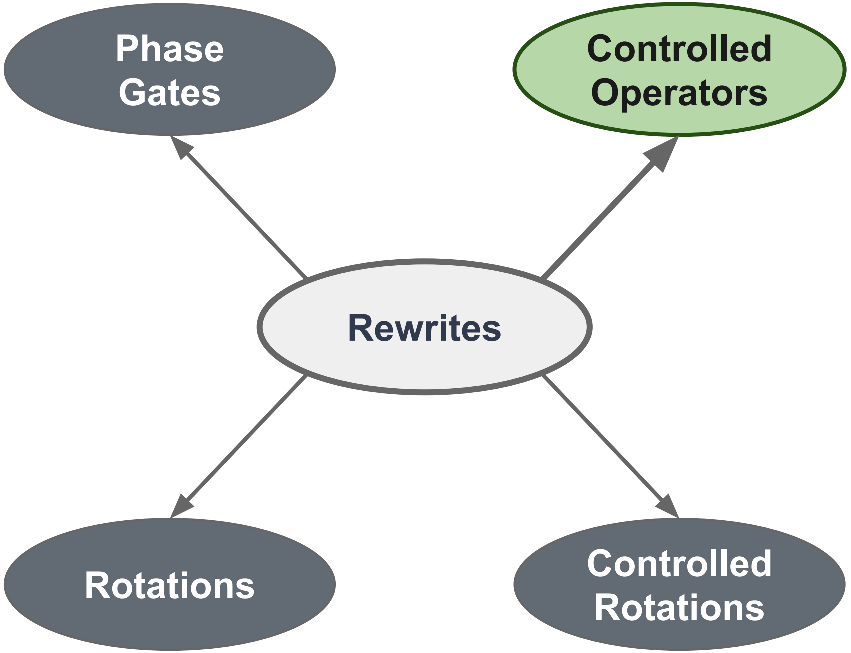


Wider Classes of Gate Decompositions





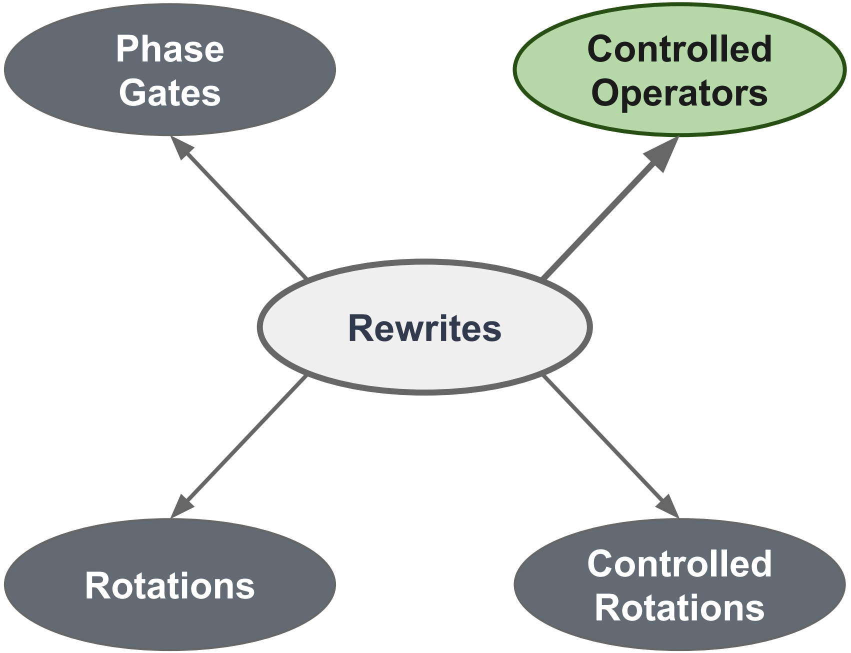
Wider Classes of Gate Decompositions



Given a unitary operator U such that:

$$U|000\rangle = |\varphi_1\rangle|0\rangle \qquad U|100\rangle = |\varphi_2\rangle|0\rangle$$
$$U|010\rangle = |\varphi_3\rangle|0\rangle \qquad U|110\rangle = |\varphi_4\rangle|1\rangle$$

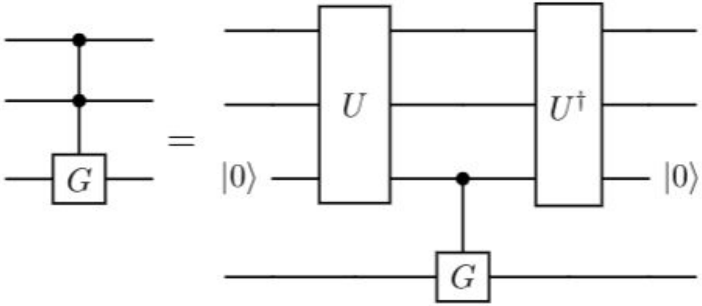
Wider Classes of Gate Decompositions



Given a unitary operator U such that:

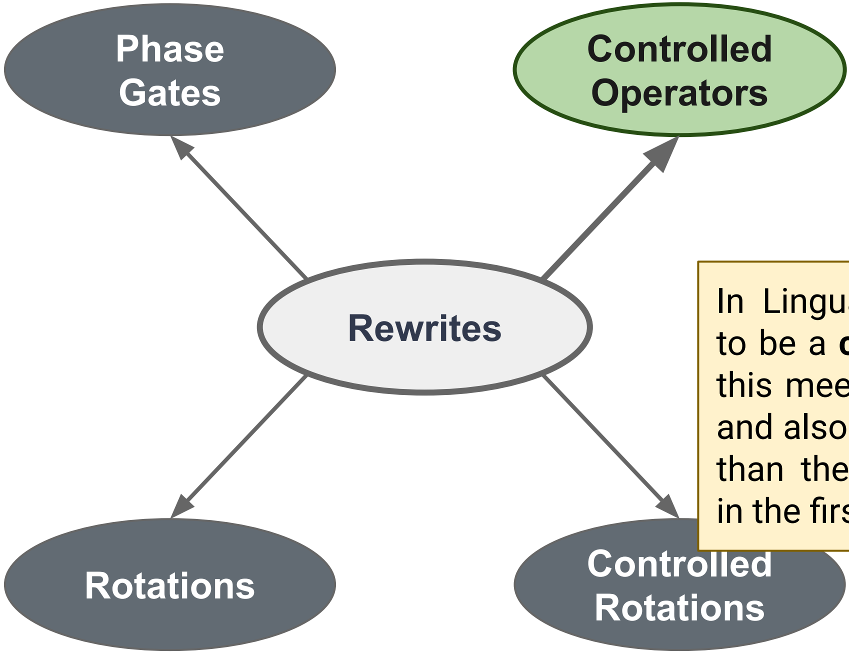
$$U|000\rangle = |\varphi_1\rangle|0\rangle \qquad U|100\rangle = |\varphi_2\rangle|0\rangle$$
$$U|010\rangle = |\varphi_3\rangle|0\rangle \qquad U|110\rangle = |\varphi_4\rangle|1\rangle$$

The following equation necessarily hold:



[35] Selinger, 2013.

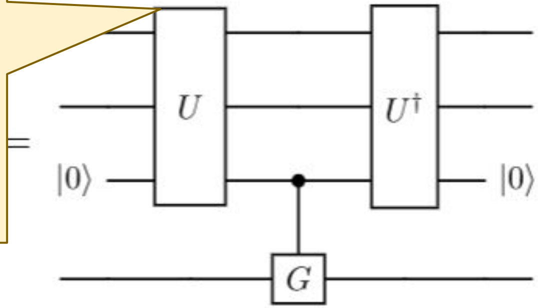
Wider Classes of Gate Decompositions



Given a unitary operator U such that:

$$U|000\rangle = |\varphi_1\rangle|0\rangle \qquad U|100\rangle = |\varphi_2\rangle|0\rangle$$
$$U|010\rangle = |\varphi_3\rangle|0\rangle \qquad U|110\rangle = |\varphi_4\rangle|1\rangle$$

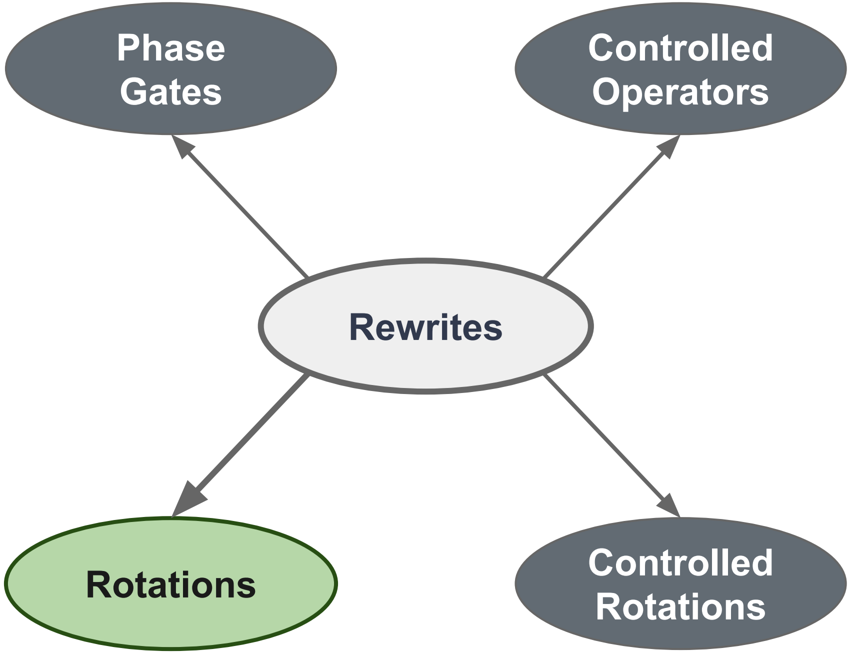
The following equation necessarily hold:



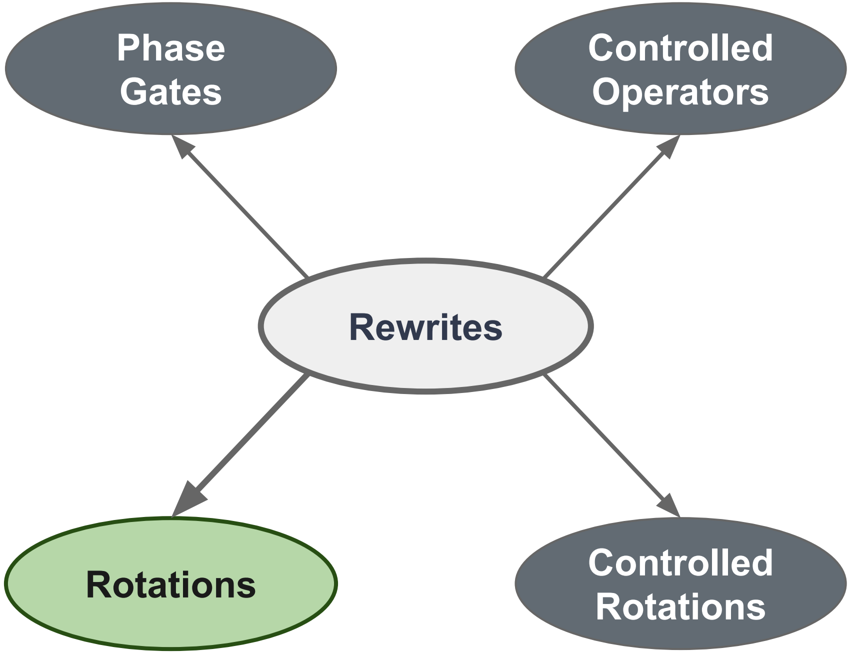
In LinguaQuanta, we take U to be a **controlled iX** gate as this meets the specifications and also has a **lower T-count** than the Toffoli gate shown in the first decomposition.

[35] Selinger, 2013.

Wider Classes of Gate Decompositions



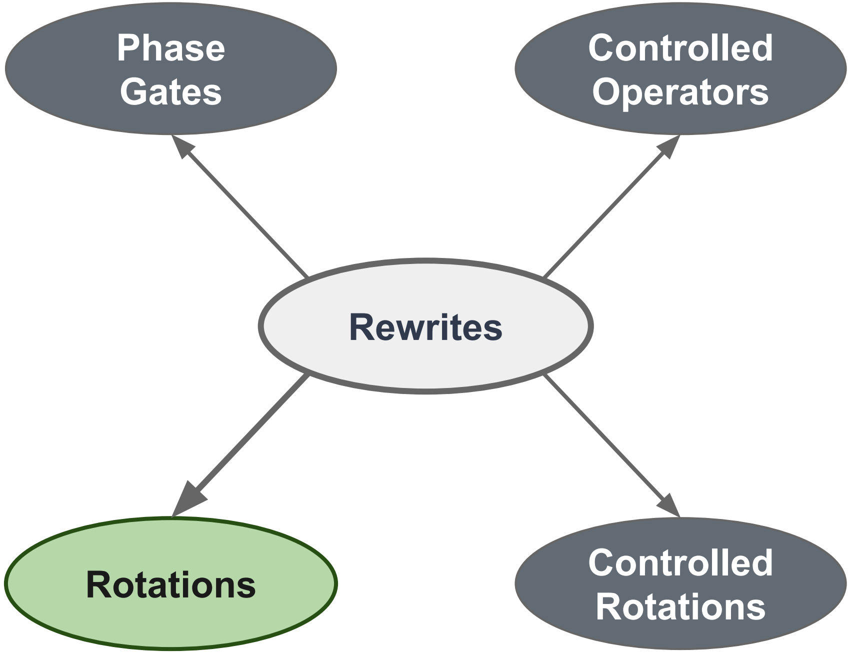
Wider Classes of Gate Decompositions



Recall the following fact about matrix exponentials.

$$e^{U^\dagger V U} = U^\dagger e^V U$$

Wider Classes of Gate Decompositions



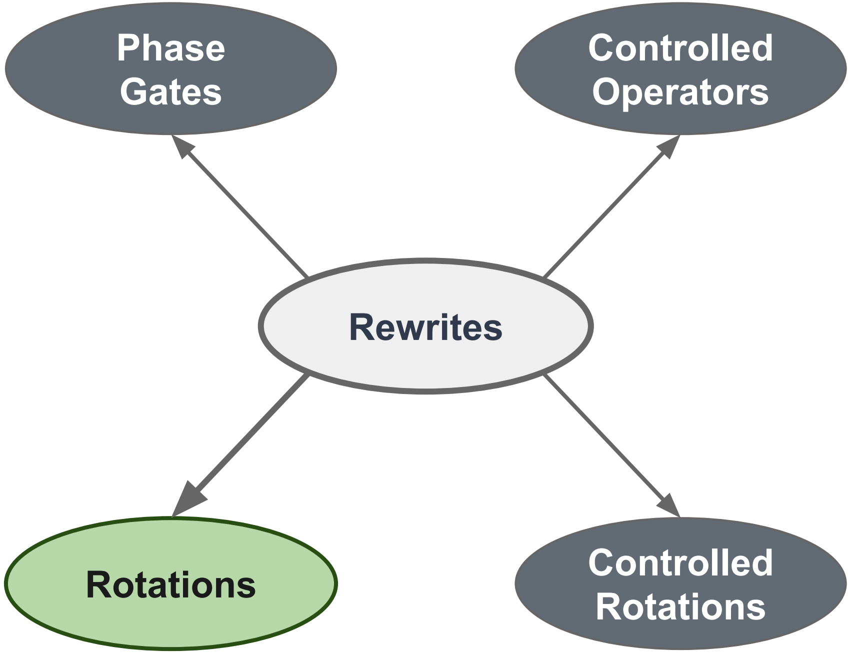
Recall the following fact about matrix exponentials.

$$e^{U^\dagger V U} = U^\dagger e^V U$$

Standard rotations are also matrix exponentials.

$$R_A(\theta) = e^{iA\theta} \qquad R_B(\theta) = e^{iB\theta}$$

Wider Classes of Gate Decompositions



Recall the following fact about matrix exponentials.

$$e^{U^\dagger V U} = U^\dagger e^V U$$

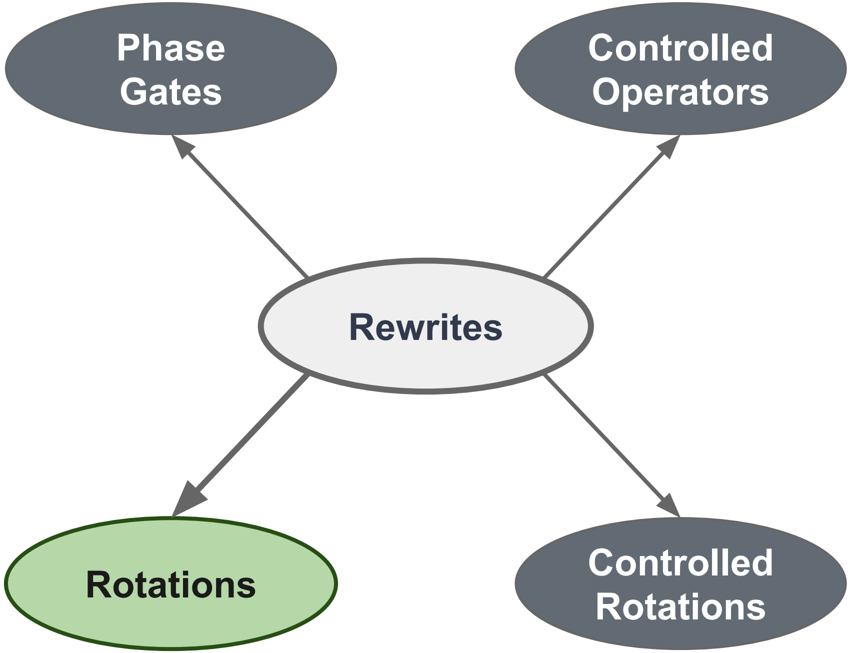
Standard rotations are also matrix exponentials.

$$R_A(\theta) = e^{iA\theta} \qquad R_B(\theta) = e^{iB\theta}$$

Let's further assume the following relation.

$$B = V^\dagger A V$$

Wider Classes of Gate Decompositions



Recall the following fact about matrix exponentials.

$$e^{U^\dagger V U} = U^\dagger e^V U$$

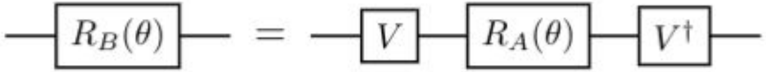
Standard rotations are also matrix exponentials.

$$R_A(\theta) = e^{iA\theta} \qquad R_B(\theta) = e^{iB\theta}$$

Let's further assume the following relation.

$$B = V^\dagger A V$$

Then we obtain the following equality.

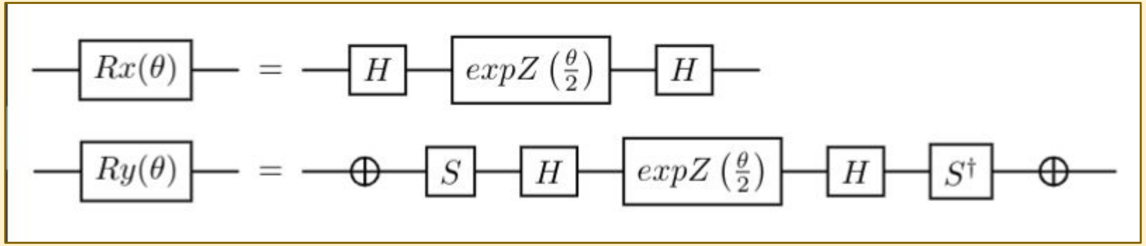


Wider Classes of Gate Decompositions

Phase

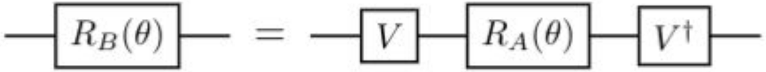
Controlled

Recall the following fact about matrix exponentials.

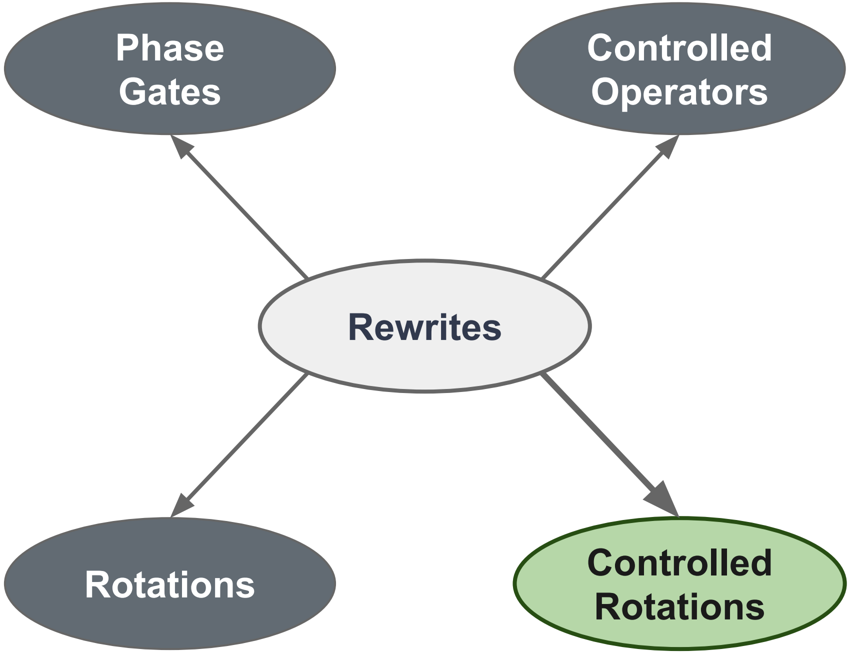


Rotations

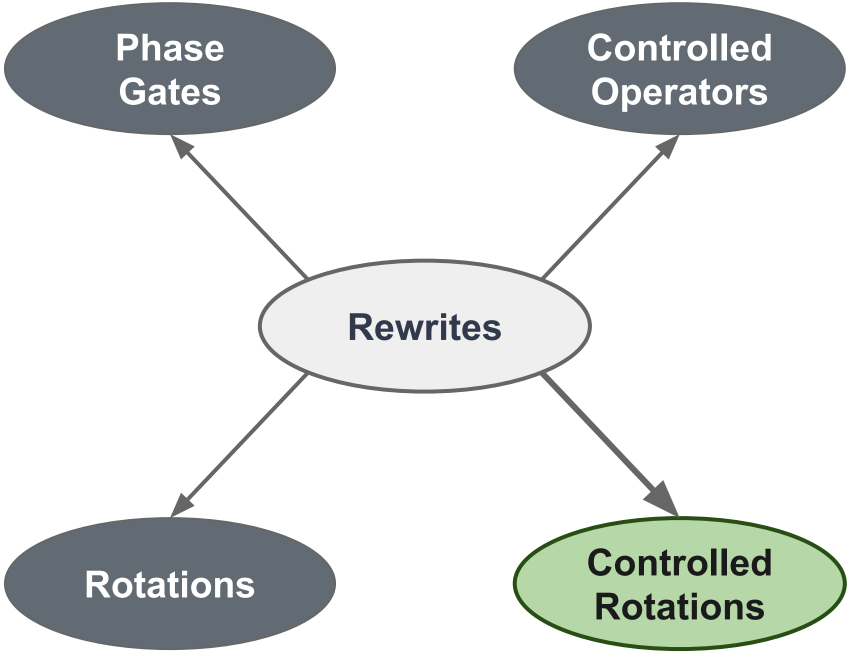
Rotations



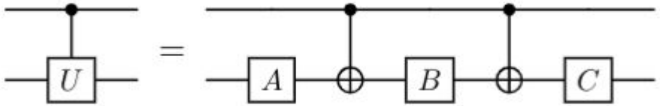
Wider Classes of Gate Decompositions



Wider Classes of Gate Decompositions

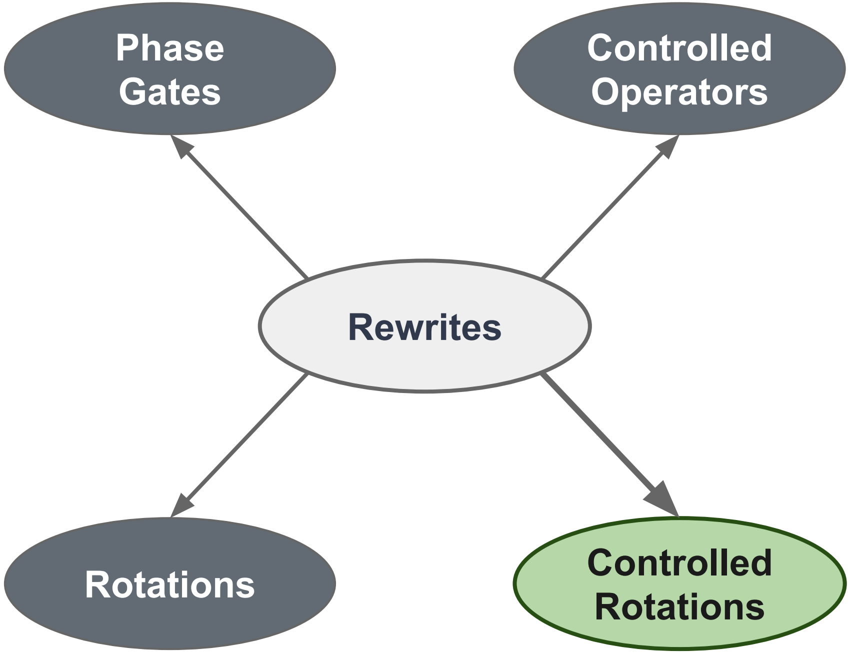


Assume that U = CXBXA and CBA = I.

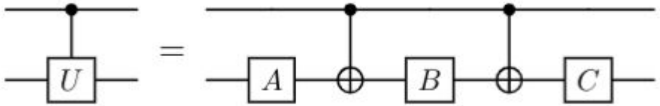


[4] A. Barenco et al., 1995.

Wider Classes of Gate Decompositions



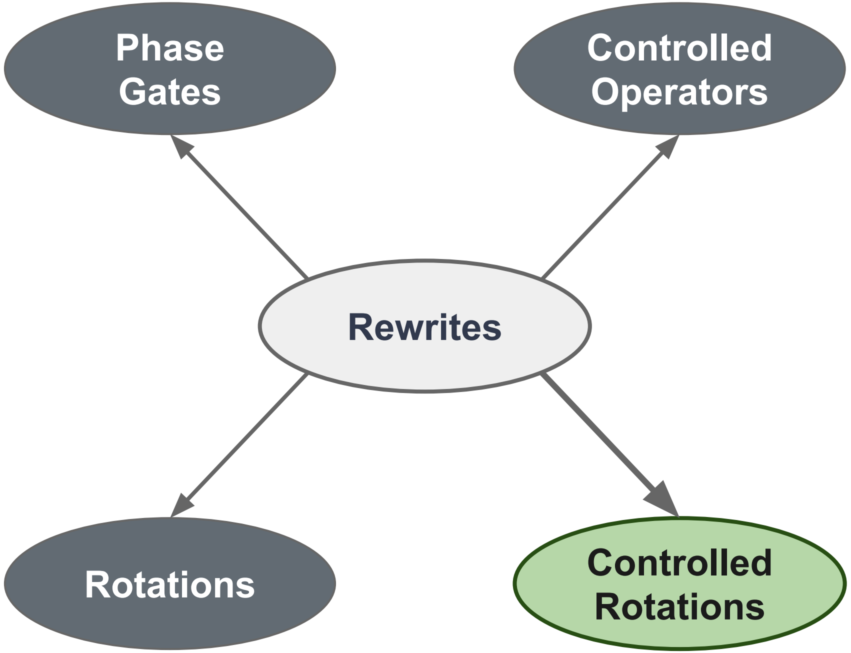
Assume that U = CXBXA and CBA = I.



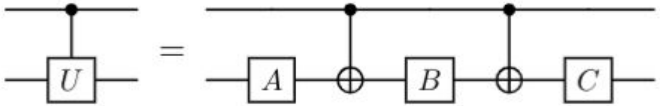
[4] A. Barenco et al., 1995.

Assume that D is self-inverse and R is a rotation satisfying (DXD)R(θ)(DXD) = R(-θ) and R(θ)R(-θ) = I.

Wider Classes of Gate Decompositions



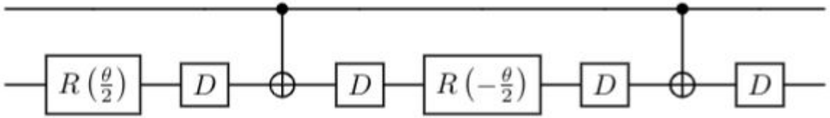
Assume that U = CXBXA and CBA = I.



[4] A. Barenco et al., 1995.

Assume that D is self-inverse and R is a rotation satisfying (DXD)R(θ)(DXD) = R(-θ) and R(θ)R(-θ) = I.

The following is a decomposition of C(R(θ)):

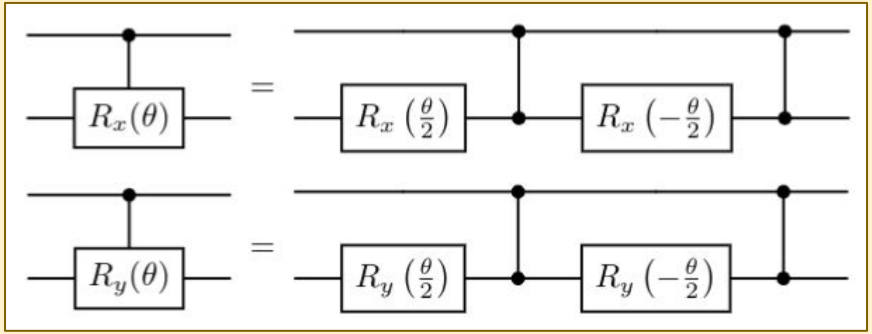


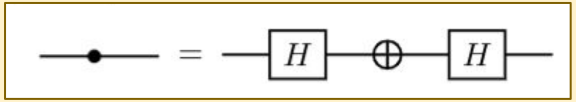
Wider Classes of Gate Decompositions

Phase

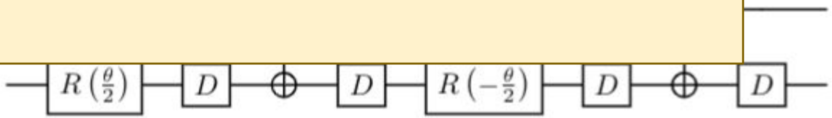
Controlled

Assume that U = CXBXA and CBA = I.





= I.



Rotations

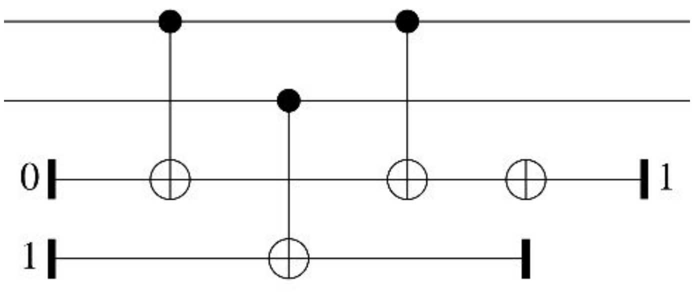
Rotations

Ancilla Management

Translation from Quipper to OpenQASM

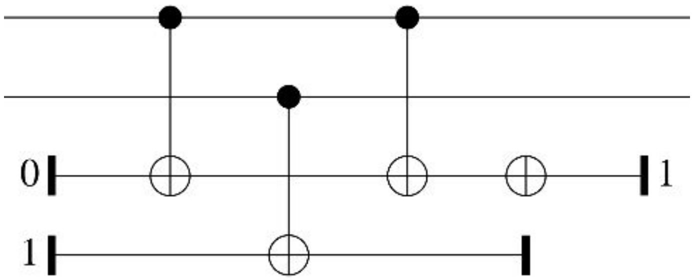
Simulating Ancillas in OpenQASM 3

Quipper circuits allow for **ancilla qubits**.



Simulating Ancillas in OpenQASM 3

Quipper circuits allow for **ancilla qubits**.

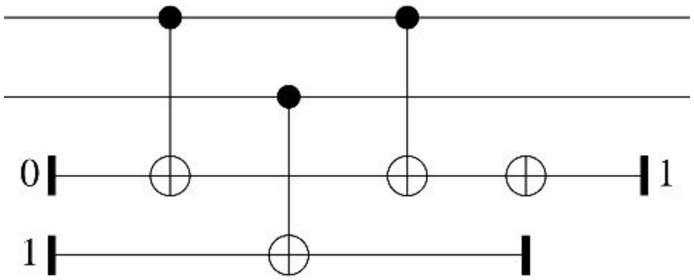


Ancillas are managed using the **following primitives**.

`Term` `Init` `Discard`

Simulating Ancillas in OpenQASM 3

Quipper circuits allow for **ancilla qubits**.



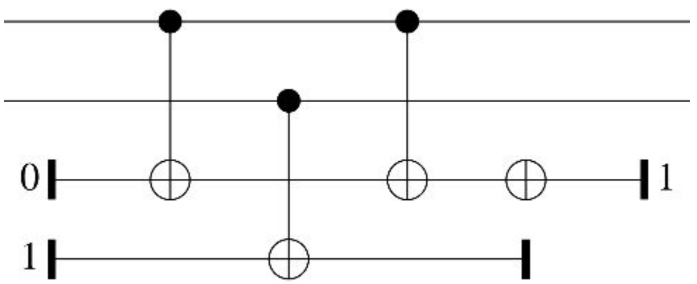
Ancillas are managed using the **following primitives**.

`Term` `Init` `Discard`

Conversely, OpenQASM allocates qubits **up front**.

Simulating Ancillas in OpenQASM 3

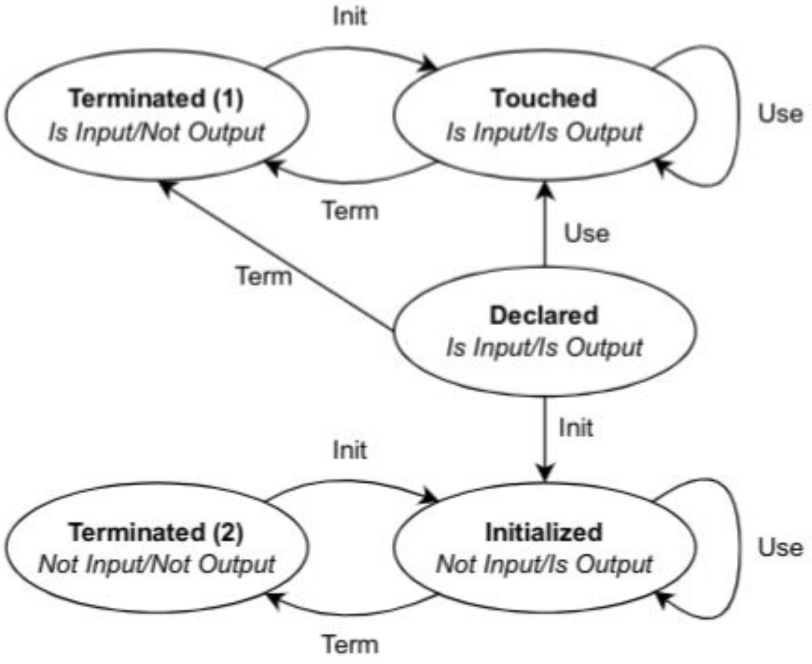
Quipper circuits allow for **ancilla qubits**.



Ancillas are managed using the **following primitives**.

`Term` `Init` `Discard`

Conversely, OpenQASM allocates qubits **up front**.



Other Features and Ongoing Work

An Advertisement for Our Upcoming Paper

See the full paper for...

1. Design features that **reduce the distance** between a round translation and the identity morphism.
2. Emulating Quipper **type conversions** in OpenQASM 3.
3. Emulating the **OpenQASM measurement semantics** in Quipper.
4. Early progress on ensuring the **correctness of pre-processing** components (control inlining).
5. A **fully-worked example** based around phase estimation.
6. A discussion surrounding the **compositionality of parameterized gates**, and a proposal for abstract types.
7. Translating **Quipper ancillas** to OpenQASM.

** The full paper if a work-in-progress, but a draft should be available in the near future. Please reach out to me!*

Limitations and Future Work

A Roadmap for LinguaQuanta

A List of Key Limitations

Ongoing Development:

1. Support for type conversion in Quipper.
2. Support for OpenQASM’s user-defined gates.
3. Support for opaque gates.
4. Basic support for classical control.
5. Constant propagation for OpenQASM 3.
6. Bounded loop unwinding for OpenQASM 3.

Out-of-scope Features:

1. No-control flags in Quipper.
2. Dynamic lifting in Quipper.
3. Generalized controls in Quipper
4. OpenPulse and physical qubits in OpenQASM 3.
5. Floats and dynamic arrays in OpenQASM 3.

Future Research Directions

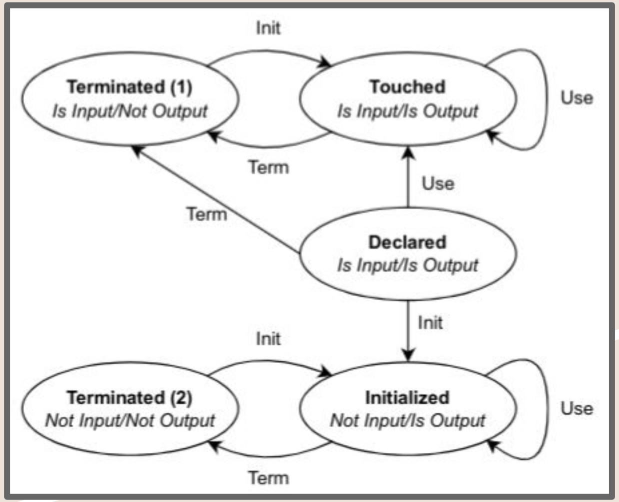
1. Categorical semantics for preprocessing in a transpilation pipeline.
2. Formal verification of LinguaQuanta and a general methodology.
3. Whitebox fuzzing of pipeline software through pipeline rewriting.
4. Empirical evaluations of LinguaQuanta (performance and conformance).

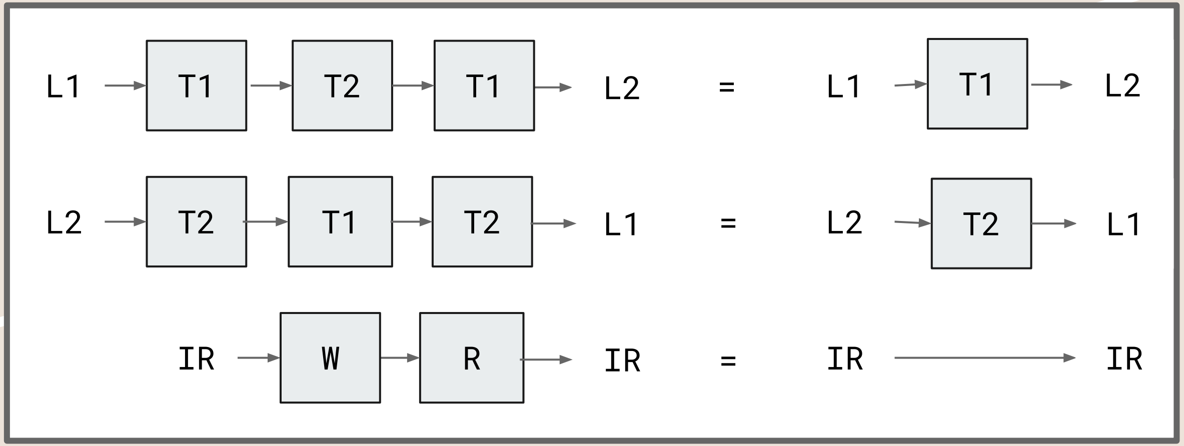
Questions?

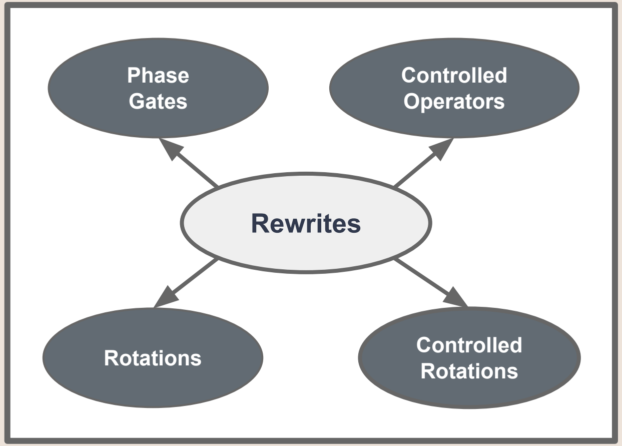
Want to learn more? Contact: scott.wesley@dal.ca

Want to try LinguaQuanta? Visit: github.com/onestruggler/qasm-quipper

A draft of the paper will be available through the IWQC website.







Appendix A

Sub-Languages, Lattices, and Pipelines

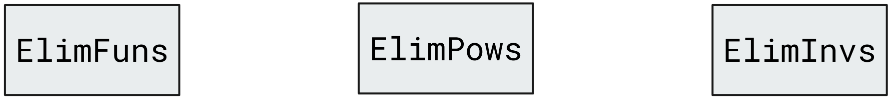
Future Work: Hierarchical Types

There are many **preprocessing** tools in LinguaQuanta.

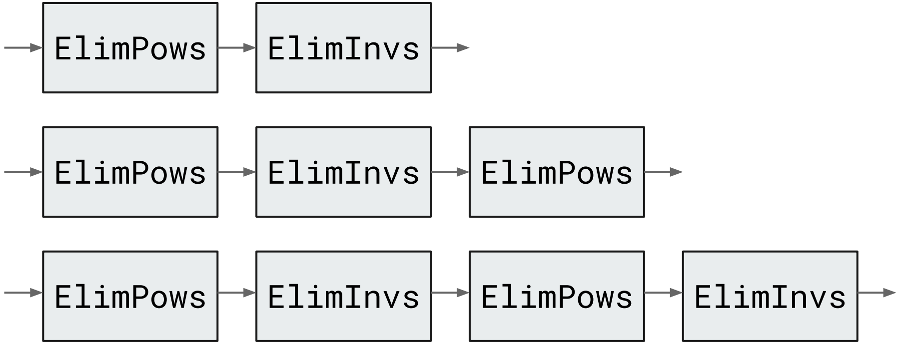
`ElimFuns` `ElimPows` `ElimInvs`

Future Work: Hierarchical Types

There are many **preprocessing** tools in LinguaQuanta.

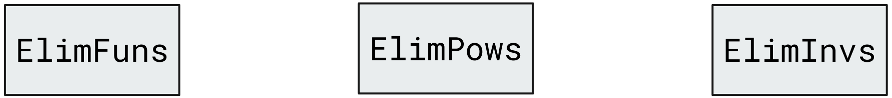


One could expect these pipelines to be **equivalent**.

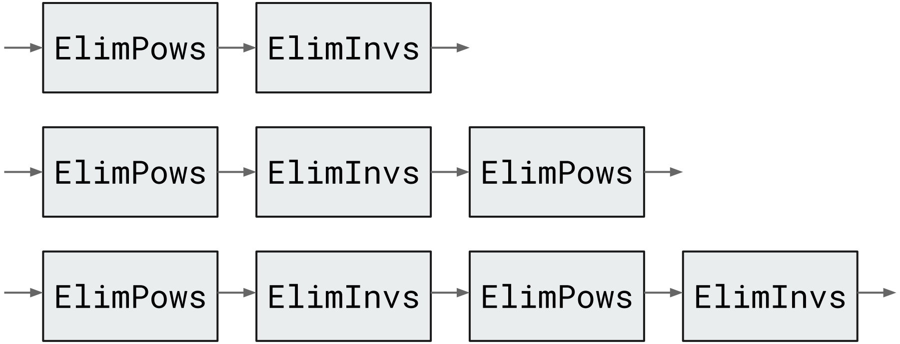


Future Work: Hierarchical Types

There are many **preprocessing** tools in LinguaQuanta.



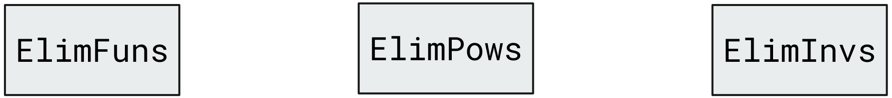
One could expect these pipelines to be **equivalent**.



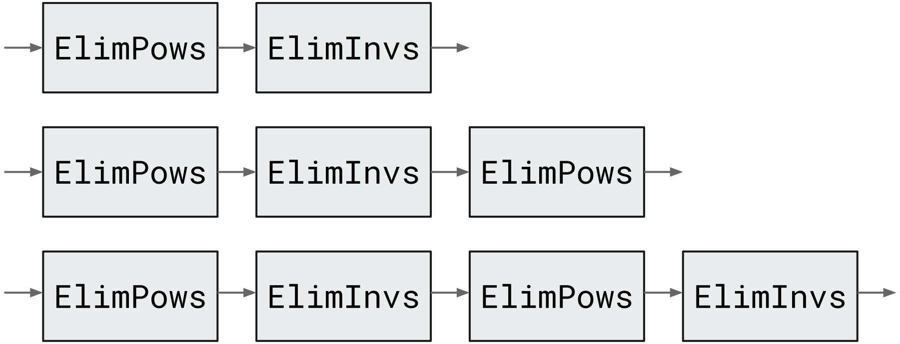
Though **commutativity** seems **too strong** a property.

Future Work: Hierarchical Types

There are many **preprocessing** tools in LinguaQuanta.

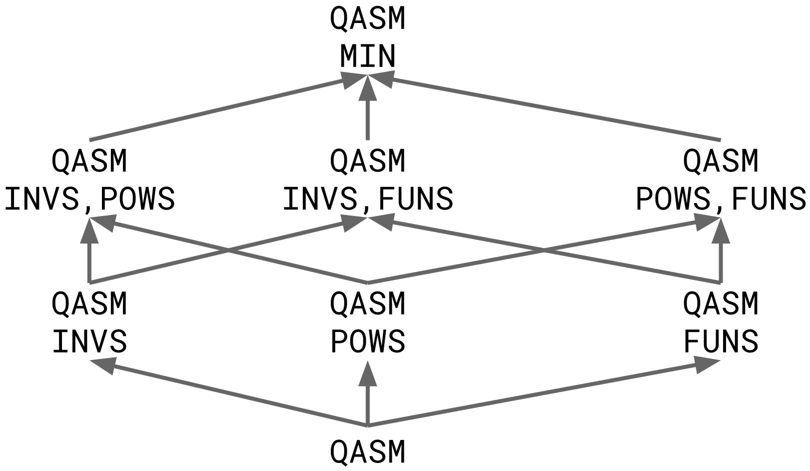


One could expect these pipelines to be **equivalent**.



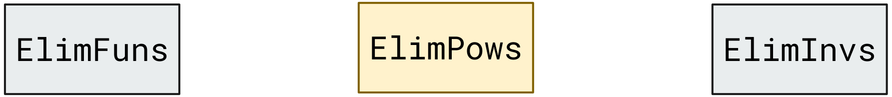
Though **commutativity** seems **too strong** a property.

Our proposed solution is some sort of **subtyping**.

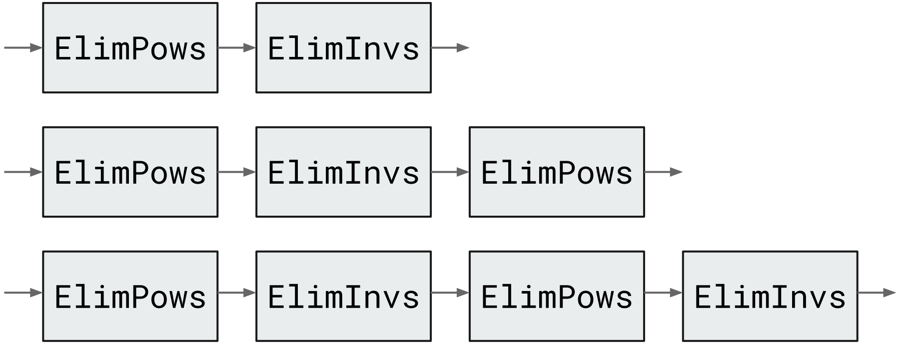


Future Work: Hierarchical Types

There are many **preprocessing** tools in LinguaQuanta.

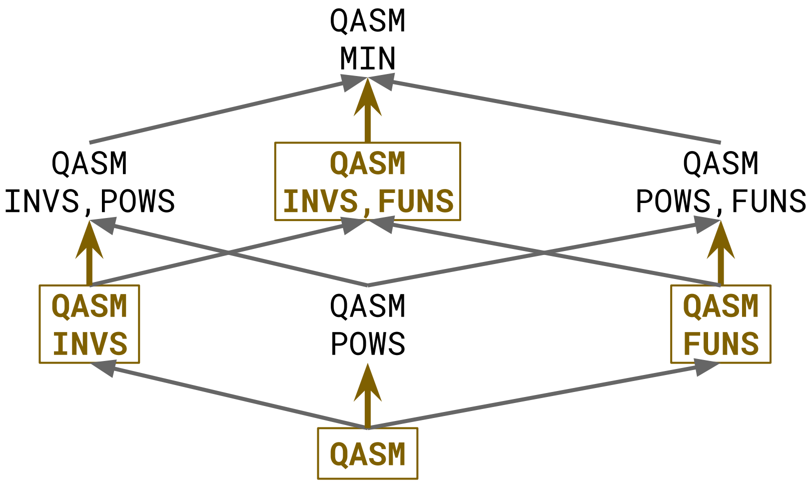


One could expect these pipelines to be **equivalent**.



Though **commutativity** seems **too strong** a property.

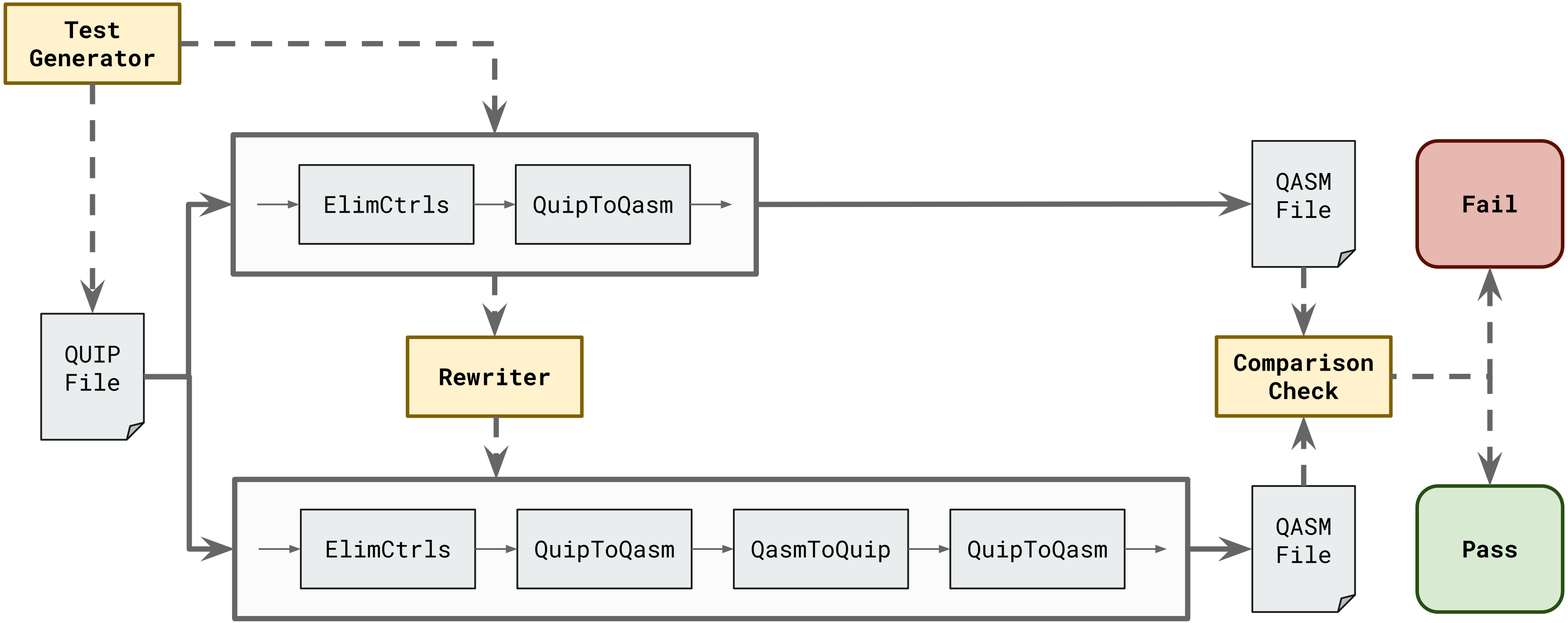
Our proposed solution is some sort of **subtyping**.



Appendix B

Whitebox Fuzzing and Pipeline Software

Future Work: Whitebox Fuzzing



Questions?

Want to learn more? Contact: scott.wesley@dal.ca

Want to try LinguaQuanta? Visit: github.com/onestruggler/qasm-quipper

A draft of the paper will be available through the IWQC website.