UNIVERSITY OF CALGARY

An investigation of the underpinnings of quantum and reversible computing

subtitle

by

Brett Gordon Giles

A DISSERTATION

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

August, 2013

Abstract

Acknowledgements

Table of Contents

List of Tables

List of Figures and Illustrations

List of Symbols, Abbreviations and Nomenclature

Symbol	Definition
U of C	University of Calgary
$\mathbb{N}$	The set of natural numbers, i.e., $\{0, 1, 2, \ldots\}$
$\mathbb{Z}$	The ring of integers numbers, i.e., $\{0, \pm 1, \pm 2, \ldots\}$
$\mathbb{C}$	The field of complex numbers

Overview of thesis chapters

Category Theory

This chapter will include a basic introduction to category theory. Specific areas introduced will included definitions of categories, natural transformations and functors. It will introduce limits and co-limits, focussing on products and co-products.

This chapter will also include an introduction to restriction categories.

Reversible Computing

This will introduce the subject of reversible computing, explaining the equivalence to standard computing at the level of Turing machines and automata. (Based on work by Bennet and Abramsky respectively). It will also provide an example of a reversible language.

Inverse categories

Inverse categories will be explored, along with some basic results regarding products and splits of categories. The inverse product will be introduced, along with the concept of a discrete inverse category and the relationship to Cartesian restriction categories.

The inverse co-product and inverse categories with inverse co-products will also be addressed in this chapter.

This is based on a paper which is in preparation (joint work with R. Cockett).

Quantum Computation

The basics of quantum computation and circuits will be introduced.

Transformations of Quantum Programs

The current understanding of how to treat iteration and folding in Quantum circuits and algorithms is somewhat lacking. This chapter will present unpublished work (done under the supervision of P. Selinger) exploring this area. It will include a treatment of necessary conditions for a quantum routine, its inputs and outputs, which would allow transforming the routine into either an iterated or folded routine. Algorithms to compute this transform are also provided.

Synthesis of Quantum transformations

This chapter will discuss the issue of gate synthesis, where an arbitrary quantum transform is to be expressed in terms of a set of base gates. The histories of both approximate and exact synthesis will be reviewed.

Then, the chapter will present an algorithm for exact synthesis of single-qubit transforms over the Clifford group, together with a normal form and characterization of these. This is based on a paper that is an extension of work done by Matsumoto and Amano and is in preparation (joint work with P. Selinger.)

Finally, we will present an algorithm for exact synthesis over the Clifford group of multi-qubit transforms and characterize those transforms that may be exactly synthesized. This based on a paper published in the journal Physical Review A (joint work with P. Selinger).

Chapter 1

Category theory

1.1 Restriction categories

Restriction categories were introduced in [19] as a convenient axiomatization of partial maps.

Definition 1.1.1. A *restriction category* is a category $\mathbb{X}$ together with a *restriction operator* on maps:

$$\frac{f : A \to B}{\overline{f} : A \to A}$$

where f is an map of $\mathbb{X}$ and A, B are objects of $\mathbb{X}$, such that the following four *restriction identities* hold, whenever the compositions[1] are defined.

$$[\boldsymbol{R.1}] \ \overline{f}f = f \qquad\qquad [\boldsymbol{R.2}] \ \overline{g}\,\overline{f} = \overline{f}\,\overline{g}$$

$$[\boldsymbol{R.3}] \ \overline{\overline{f}g} = \overline{f}\,\overline{g} \qquad\qquad [\boldsymbol{R.4}] \ f\overline{g} = \overline{fg}f$$

Definition 1.1.2. A *restriction functor* is a functor which preserves the restriction. That is, given a functor $F : \mathbb{X} \to \mathbb{Y}$ with $\mathbb{X}$ and $\mathbb{Y}$ restriction categories, F is a restriction functor if:

$$F(\overline{f}) = \overline{F(f)}.$$

Any map such that $r = \overline{r}$ is an idempotent, as $\overline{r}\,\overline{r} = \overline{\overline{r}}\,\overline{r} = \overline{r}$, and is called a *restriction idempotent*. All maps $\overline{f}$ are restriction idempotents as $\overline{f} = \overline{\overline{f}}$. Below, we record some basic facts for restriction categories shown in [19] pp 4-5:

Lemma 1.1.3. *In a restriction category $\mathbb{X}$,*

---

[1]Note that composition is written in diagrammatic order throughout this paper.

(i) $\overline{f}$ is idempotent;

(v) $\overline{f}\,\overline{g} = \overline{\overline{f}\,\overline{g}}$;

(ii) $\overline{f}\overline{g} = \overline{f}\overline{g}\,\overline{f}$;

(vi) f monic implies $\overline{f} = 1$;

(iii) $\overline{f}\overline{g} = \overline{f}\overline{g}$;

(vii) $f = \overline{g}f \implies \overline{g}\,\overline{f} = \overline{f}$.

(iv) $\overline{\overline{f}} = \overline{f}$;

A map $f : A \to B$ in a restriction category is said to be *total* when $\overline{f} = 1_A$. The total maps in a restriction category form a subcategory $Total(\mathbb{X}) \subseteq \mathbb{X}$.

An example of a restriction category is PAR, the category with objects sets and arrows the partial functions between sets. In PAR, the restriction of $f : A \to B$ is:

$$
\overline{f}(x) = \begin{cases} x & \text{if } f(x) \text{ is defined,} \\ \uparrow & \text{if } f(x) \text{ is } \uparrow . \end{cases}
$$

(The symbol $\uparrow$ means that the function is undefined at that element). In PAR, the total maps correspond precisely to the functions that are defined on all elements of the domain.

1.1.1 Enrichment and meets

In any restriction category, there is a partialorder on each hom-set, given by $f \leq g$ iff $\overline{f}g = f$, where $f, g : A \to B$.

Lemma 1.1.4. *In a restriction category $\mathbb{X}$:*

(i) $\leq$ *as defined above is a partial order on each hom-set;*

(ii) $f \leq g \implies \overline{f} \leq \overline{g}$;

(iii) $f \leq g \implies hf \leq hg$;

(iv) $f \leq g \implies fh \leq gh$;

(v) $f \leq 1 \iff f = \overline{f}$.

Proof.

4

(i) With f, g, h parallel maps in $\mathbb{X}$, each of the requirements for a partial order is shown below:

Reflexivity: $\overline{f}f = f$ and therefore, $f \leq f$.

Anti-Symmetry: Given $\overline{f}g = f$ and $\overline{g}f = g$, it follows:

$$f = \overline{f}f = \overline{\overline{f}g}f = \overline{f}\,\overline{g}f = \overline{g}\overline{f}f = \overline{g}f = g.$$

Transitivity: Given $f \leq g$ and $g \leq h$,

$$\overline{f}h = \overline{\overline{f}g}h = \overline{f}\,\overline{g}h = \overline{f}g = f$$

showing that $f \leq h$.

(ii) The premise is that $\overline{f}g = f$. From this, $\overline{f}\,\overline{g} = \overline{\overline{f}g} = \overline{f}$, showing $\overline{f} \leq \overline{g}$.

(iii) $\overline{h\overline{f}}hg = h\overline{f}g = hf$ and therefore $hf \leq hg$.

(iv) $\overline{f}g = f$, this shows $\overline{f\overline{h}}gh = \overline{\overline{f}gh}gh = \overline{f}\,\overline{gh}gh = \overline{f}gh = fh$ and therefore $fh \leq gh$.

(v) As $f \leq 1$ means precisely $\overline{f}1 = f$.

$\square$

Lemma 1.1.4 on the preceding page shows that restriction categories are enriched in partial orders.

Definition 1.1.5. A restriction category has *meets* if there is an operation $\cap$ on parallel maps:

$$\frac{A \overset{f}{\underset{g}{\rightrightarrows}} B}{A \xrightarrow{f \cap g} B}$$

such that $f \cap g \leq f, f \cap g \leq g, f \cap f = f, h(f \cap g) = hf \cap hg$.

Meets were introduced in [9]. The following are basic results on meets:

Lemma 1.1.6. *In a restriction category* $\mathbb{X}$ *with meets, where* f, g, h *are maps in* $\mathbb{X}$, *the following are true:*

(i) $f \leq g$ *and* $f \leq h \iff f \leq g \cap h$;

(ii) $f \cap g = g \cap f$;

(iii) $\overline{f \cap 1} = f \cap 1$;

(iv) $(f \cap g) \cap h = f \cap (g \cap h)$;

(v) $r(f \cap g) = rf \cap g$ *where* $r = \overline{r}$ *is a restriction idempotent;*

(vi) $(f \cap g)r = fr \cap g$ *where* $r = \overline{r}$ *is a restriction idempotent;*

(vii) $\overline{f \cap g} \leq \overline{f}$ *(and therefore* $\overline{f \cap g} \leq \overline{g}$*);*

(viii) $(f \cap 1)f = f \cap 1$;

(ix) $e(e \cap 1) = e$ *where* e *is idempotent.*

Proof.

(i) $f \leq g$ and $f \leq h$ means precisely $f = \overline{f}g$ and $f = \overline{f}h$. Therefore,

$$\overline{f}(g \cap h) = \overline{f}g \cap \overline{f}h = f \cap f = f$$

and so $f \leq g \cap h$. Conversely, given $f \leq g \cap h$, we have $f = \overline{f}(g \cap h) = \overline{f}g \cap \overline{f}h \leq \overline{f}g$. But $f \leq \overline{f}g$ means $f = \overline{f}\,\overline{f}g = \overline{f}g$ and therefore $f \leq g$. Similarly, $f \leq h$.

(ii) From (i), as by definition, $f \cap g \leq g$ and $f \cap g \leq f$.

(iii) $f \cap 1 = \overline{f \cap 1}(f \cap 1) = (\overline{f \cap 1}f) \cap (\overline{f \cap 1}) \leq \overline{f \cap 1}$ from which the result follows.

(iv) By definition and transitivity, $(f \cap g) \cap h \leq f, g, h$ therefore by (i) $(f \cap g) \cap h \leq f \cap (g \cap h)$. Similarly, $f \cap (g \cap h) \leq (f \cap g) \cap h$ giving the equality.

6

(v) Given $rf \cap g \leq rf$, calculate:

$$rf \cap g = \overline{rf \cap g}\,rf = \overline{r(rf \cap g)}f = \overline{rrf \cap rg}f = \overline{r(f \cap g)}f = r\overline{f \cap g}f = r(f \cap g).$$

(vi) Using the previous point with the restriction idempotent $\overline{fr}$,

$$fr \cap g = f\overline{r} \cap g = \overline{fr}f \cap g = \overline{fr}(f \cap g) = \overline{fr}\,\overline{f \cap g}f$$

$$= \overline{f \cap g}\,\overline{fr}f = \overline{f \cap g}f\overline{r} = (f \cap g)r.$$

(vii) For the first claim,

$$\overline{f \cap g}\,\overline{f} = \overline{\overline{f}(f \cap g)} = \overline{(\overline{f}f) \cap g} = \overline{f \cap g}.$$

The second claim then follows by (ii) on the previous page.

(viii) Given $f \cap 1 \leq f$:

$$f \cap 1 \leq f \iff \overline{f \cap 1}f = f \cap 1 \iff (f \cap 1)f = f \cap 1$$

where the last step is by item (iii) on the preceding page of this lemma.

(ix) As e is idempotent, $e(e \cap 1) = (ee \cap e) = e$.

$\square$

1.1.2 Partial monics, sections and isomorphisms

Partial isomorphisms play a central role in this paper and below we develop some their basic properties.

Definition 1.1.7. A map f in a restriction category $\mathbb{X}$ is said:

- To be a *partial isomorphism* when there is a *partial inverse*, written $f^{(-1)}$ with $ff^{(-1)} = \overline{f}$ and $f^{(-1)}f = \overline{f^{(-1)}}$;

- To be a *partial monic* if $hf = kf \implies h\overline{f} = k\overline{f}$;

- To be a *partial section* if there exists an h such that $fh = \overline{f}$;

7

- To be a *restriction monic* if it is a section s with a retraction r such that $rs = \overline{rs}$.

Lemma 1.1.8. *In a restriction category:*

    (i) f, g partial monic implies fg is partial monic;

    (ii) f a partial section implies f is partial monic;

    (iii) f, g partial sections implies fg is a partial section;

    (iv) The partial inverse of f, when it exists, is unique;

    (v) If f, g have partial inverses and fg exists, then fg has a partial inverse;

    (vi) A restriction monic s is a partial isomorphism.

Proof.

(i) Suppose $hfg = kfg$. As g is partial monic, $hf\overline{g} = kf\overline{g}$. Therefore:

$$h\overline{fg}f = k\overline{fg}f \qquad\qquad [\boldsymbol{R.4}]$$

$$h\overline{fg}\,\overline{f} = k\overline{fg}\,\overline{f} \qquad\qquad f\text{partial monic}$$

$$h\overline{fg} = k\overline{fg} \qquad\qquad \text{Lemma } 1.1.3, \text{(ii)}$$

(ii) Suppose $gf = kf$. Then, $g\overline{f} = gfh = kfh = k\overline{f}$.

(iii) We have $fh = \overline{f}$ and $gh' = \overline{g}$. Therefore,

$$fgh'h = f\overline{g}h \qquad\qquad g \text{ partial section}$$

$$= \overline{fg}fh \qquad\qquad [\boldsymbol{R.4}]$$

$$= \overline{fg}\,\overline{f} \qquad\qquad f \text{ partial section}$$

$$= \overline{f}\,\overline{fg} \qquad\qquad [\boldsymbol{R.2}]$$

$$= \overline{\overline{f}fg} \qquad\qquad [\boldsymbol{R.3}]$$

$$= \overline{fg} \qquad\qquad [\boldsymbol{R.1}]$$

(iv) Suppose both $f^{(-1)}$ and f^* are partial inverses of f. Then,

$$f^{(-1)} = \overline{f^{(-1)}} f^{(-1)} = f^{(-1)} f f^{(-1)} = f^{(-1)} \overline{f} = f^{(-1)} f f^* = f^{(-1)} f \overline{f^*} f^*$$

$$= \overline{f^{(-1)} \overline{f^*}} f^* = \overline{f^*} \overline{f^{(-1)}} f^* = f^* f \overline{f^{(-1)}} f^* = f^* f f^{(-1)} f f^* = f^* f f^* = f^*$$

(v) For $f : A \to B$, $g : B \to C$ with partial inverses $f^{(-1)}$ and $g^{(-1)}$ respectively, the partial inverse of fg is $g^{(-1)} f^{(-1)}$. Calculating $fg g^{(-1)} f^{(-1)}$ using all the restriction identities:

$$f g g^{(-1)} f^{(-1)} = f \overline{g} f^{(-1)} = \overline{fg} f f^{(-1)} = \overline{fg} \overline{f} = \overline{f} \overline{fg} = \overline{\overline{f} fg} = \overline{fg}.$$

The calculation of $g^{(-1)} f^{(-1)} fg = \overline{g^{(-1)} f^{(-1)}}$ is similar.

(vi) The partial inverse of s is $\overline{rs}\, r$. First, note that $\overline{\overline{rs}\, r} = \overline{rs}\, \overline{r} = \overline{r}\, \overline{rs} = \overline{\overline{r}\, rs} = \overline{rs}$. Then, it follows that $(\overline{rs}\, r)s = rs = \overline{rs} = \overline{rs}r$ and $s(\overline{rs}\, r) = sr\overline{s} = \overline{s}$.

$\square$

A restriction category in which every map is a partial isomorphism is called an *inverse category*.

An interesting property of inverse categories:

Lemma 1.1.9. *In an inverse category, all idempotents are restriction idempotents.*

Proof. Given an idempotent e,

$$\overline{e} = e e^{(-1)} = e e e^{(-1)} = e \overline{e} = \overline{e} \overline{e} e = \overline{e} e = e.$$

$\square$

1.1.3 Split restriction categories

The split restriction category, $\mathrm{K}_E(\mathbb{X})$ is defined as:

Objects: (A, e), where A is an object of $\mathbb{X}$, $e : A \to A$ and $e \in E$.

Maps: $f : (A, d) \to (B, e)$ is given by $f : A \to B$ in $\mathbb{X}$, where $f = dfe$.

Identity: The map e for (A, e).

Composition: inherited from $\mathbb{X}$.

This is the standard idempotent splitting construction, also known as the Karoubi envelope.

Note that for $f : (A, d) \to (B, e)$, by definition, in $\mathbb{X}$ we have $f = dfe$, giving

$$df = d(dfe) = ddfe = dfe = f \quad \text{and} \quad fe = (dfe)e = dfee = dfe = f.$$

When $\mathbb{X}$ is a restriction category, there is an immediate candidate for a restriction in $K_E(\mathbb{X})$. If $f \in K_E(\mathbb{X})$ is $e_1 f e_2$ in $\mathbb{X}$, then define $\overline{f}$ as given by $e_1 \overline{f}$ in $\mathbb{X}$. Note that for $f : (A, d) \to (B, e)$, in $\mathbb{X}$ we have:

$$d\overline{f} = \overline{df}d = \overline{f}d.$$

Proposition 1.1.10. *If $\mathbb{X}$ is a restriction category and E is a set of idempotents, then the restriction as defined above makes $K_E(\mathbb{X})$ a restriction category.*

Proof. The restriction takes $f : (A, e_1) \to (B, e_2)$ to an endomorphism of (A, e_1). The restriction is in $K_E(\mathbb{X})$ as

$$e_1(e_1\overline{f})e_1 = e_1\overline{f}e_1 = \overline{e_1 f}e_1 e_1 = \overline{e_1 f}e_1 = e_1\overline{f}.$$

Checking the 4 restriction axioms:

[**R.1**] $[\![\overline{f}f]\!] = e_1\overline{f}f = e_1 f = [\![f]\!]$

[**R.2**] $[\![\overline{g}\overline{f}]\!] = e_1\overline{g}e_1\overline{f} = e_1 e_1\overline{g}\overline{f} = e_1 e_1\overline{f}\overline{g} = e_1\overline{f}e_1\overline{g} = [\![\overline{f}\overline{g}]\!]$

[**R.3**] $[\![\overline{\overline{f}g}]\!] \equiv e_1\overline{e_1\overline{f}g} = \overline{e_1 e_1\overline{f}g}e_1 = \overline{e_1\overline{f}g}e_1 = e_1\overline{\overline{f}g} = e_1\overline{f}\overline{g} = e_1 e_1\overline{f}\overline{g} = e_1\overline{f}e_1\overline{g} = [\![\overline{f}\ \overline{g}]\!]$

[**R.4**] $[\![f\overline{g}]\!] = e_1 f e_2\overline{g} = \overline{e_1 f e_2 g}e_1 f e_2 = \overline{e_1 e_1 f e_2 g}e_1 e_1 f e_2$

$$= e_1\overline{e_1 f e_2 g}e_1 f e_2 = e_1\overline{f g}e_1 f e_2 = [\![\overline{fg}f]\!]$$

$\square$

10

Given this, provided all identity maps are in E, $K_E(\mathbb{X})$ is a restriction category with $\mathbb{X}$ as a full sub-restriction category, via the embedding defined by taking an object A in $\mathbb{X}$ to the object $(A, 1)$ in $K_E(\mathbb{X})$. Furthermore, the property of being an inverse category is preserved by splitting.

Lemma 1.1.11. *When $\mathbb{X}$ is an inverse category, $K_E(X)$ is an inverse category.*

Proof. The inverse of $f : (A, e_1) \to (B, e_2)$ in $K_E(\mathbb{X})$ is $e_2 f^{(-1)} e_1$ as

$$[\![f f^{(-1)}]\!] = e_1 f e_2 e_2 f^{(-1)} e_1 = e_1 e_1 f e_2 f^{(-1)} e_1 = e_1 f f^{(-1)} e_1 = e_1 e_1 \overline{f} e_1 = e_1 \overline{f} = [\![\overline{f}]\!]$$

and

$$[\![f^{(-1)} f]\!] = e_2 f^{(-1)} e_1 e_1 f e_2 = e_2 f^{(-1)} e_1 f e_2 e_2 = e_2 f^{(-1)} f e_2$$

$$= e_2 e_2 \overline{f^{(-1)}} e_2 = e_2 \overline{f^{(-1)}} = [\![\overline{f^{(-1)}}]\!]$$

$\square$

Proposition 1.1.12. *In a restriction category $\mathbb{X}$, with meets, let R be the set of restriction idempotents. Then, $K(\mathbb{X}) \cong K_R(\mathbb{X})$ (where $K(\mathbb{X})$ is the split of $\mathbb{X}$ over all idempotents). Furthermore, $K_R(\mathbb{X})$ has meets.*

Proof. The proof below first shows the equivalence of the two categories, then addresses the claim that $K_R(\mathbb{X})$ has meets.

For equivalence, we require two functors,

$$U : K_R(\mathbb{X}) \to K(\mathbb{X}) \text{ and } V : K(\mathbb{X}) \to K_R(\mathbb{X}),$$

with:

$$UV \cong I_{K_R(\mathbb{X})} \tag{1.1}$$

$$VU \cong I_{K(\mathbb{X})}. \tag{1.2}$$

11

U is the standard inclusion functor. V will take the object (A, e) to $(A, e \cap 1)$ and the map $f : (A, e_1) \to (B, e_2)$ to $(e_1 \cap 1)f$.

V is a functor as:

Well Defined: If $f : (A, e_1) \to (B, e_2)$, then $(e_1 \cap 1)f$ is a map in $\mathbb{X}$ from A to B and

$$(e_1 \cap 1)(e_1 \cap 1)f(e_2 \cap 1) = (e_1 \cap 1)(fe_2 \cap f) = (e_1 \cap 1)(f \cap f) = (e_1 \cap 1)f,$$

therefore, $V(f) : V((A, e_1)) \to V((B, e_2))$.

Identities: $V(e) = (e \cap 1)e = e \cap 1$ by lemma 1.1.6 on page 6.

Composition: $V(f)V(g) = (e_1 \cap 1)f(e_2 \cap 1)g = (e_1 \cap 1)fe_2(e_2 \cap 1)g = (e_1 \cap 1)f(e_2 \cap e_2)g =$
$$(e_1 \cap 1)fe_2g = (e_1 \cap 1)fg = V(fg).$$

Recalling from Lemma 1.1.6 on page 6, $(e \cap 1)$ is a restriction idempotent. Using this fact, the commutativity of restriction idempotents and the general idempotent identities from 1.1.6 on page 6, the composite functor UV is the identity on $K_r(\mathbb{X})$ as when e is a restriction idempotent, $e = e(e \cap 1) = (e \cap 1)e = (e \cap 1)$.

For the other direction, note that for a particular idempotent $e : A \to A$, this gives the maps $e : (A, e) \to (A, e \cap 1)$ and $e \cap 1 : (A, e \cap 1) \to (A, e)$, again by 1.1.6 on page 6. These maps give the natural isomorphism between I and VU as

$$\begin{array}{ccc} (A, e) \xrightarrow{\ e\ } (A, e \cap 1) \\ \searrow_{e} \quad \downarrow^{e \cap 1} \\ (A, e) \end{array} \qquad \text{and} \qquad \begin{array}{ccc} (A, e \cap 1) \xrightarrow{\ e \cap 1\ } (A, e) \\ \searrow_{e \cap 1} \quad \downarrow^{e} \\ (A, e \cap 1) \end{array}$$

both commute. Therefore, $UV = I$ and $VU \cong I$, giving an equivalence of the categories.

For the rest of this proof, the bolded functions, e.g., $\mathbf{f}$ are in $K_R(\mathbb{X})$. Italic functions, e.g., f are in $\mathbb{X}$.

To show that $K_R(\mathbb{X})$ has meets, designate the meet in $K_R(\mathbb{X})$ as $\cap_K$ and define $\mathbf{f} \cap_K \mathbf{g}$ as the map given by the $\mathbb{X}$ map $f \cap g$, where $\mathbf{f}, \mathbf{g} : (A, d) \to (B, e)$ in $K_R(\mathbb{X})$ and $f, g : A \to B$ in $\mathbb{X}$. This is a map in $K_R(\mathbb{X})$ as $d(f \cap g)e = (df \cap dg)e = (f \cap g)e = (fe \cap g) = f \cap g$ where the penultimate equality is by 1.1.6 on page 6. By definition $\overline{\mathbf{f} \cap_K \mathbf{g}}$ is $\overline{df \cap g}$.

It is necessary to show $\cap_{\mathrm{K}}$ satisfies the four meet properties.

- $\mathbf{f} \cap_{\mathrm{K}} \mathbf{g} \leq \mathbf{f}$: We need to show $\overline{\mathbf{f} \cap_{\mathrm{K}} \mathbf{g}}\mathbf{f} = \mathbf{f} \cap_{\mathrm{K}} \mathbf{g}$. Calculating now in $\mathbb{X}$:

$$\overline{df \cap g}f = \overline{d(f \cap g)}df$$
$$= \overline{df \cap dg}df$$
$$= \overline{f \cap g}f$$
$$= f \cap g$$

which is the definition of $\mathbf{f} \cap_{\mathrm{K}} \mathbf{g}$.

- $\mathbf{f} \cap_{\mathrm{K}} \mathbf{g} \leq \mathbf{g}$: Similarly and once again calculating in $\mathbb{X}$,

$$\overline{df \cap g}g = \overline{d(f \cap g)}dg$$
$$= \overline{df \cap dg}dg$$
$$= \overline{f \cap g}g$$
$$= f \cap g$$

which is the definition of $\mathbf{f} \cap_{\mathrm{K}} \mathbf{g}$.

- $\mathbf{f} \cap_{\mathrm{K}} \mathbf{f} = \mathbf{f}$: From the definition, this is $f \cap f = f$ which is just $\mathbf{f}$.

- $\mathbf{h}(\mathbf{f} \cap_{\mathrm{K}} \mathbf{g}) = \mathbf{hf} \cap_{\mathrm{K}} \mathbf{hg}$: From the definition, this is given in $\mathbb{X}$ by $h(f \cap g) = hf \cap hg$ which in $\mathrm{K}_R(\mathbb{X})$ is $\mathbf{hf} \cap_{\mathrm{K}} \mathbf{hg}$.

$\square$

1.1.4 Partial Map Categories

In [19], it is shown that split restriction categories are equivalent to *partial map categories*. The main definitions and results related to partial map categories are given below.

Definition 1.1.13. A collection $\mathcal{M}$ of monics is *a stable system of monics* when it includes all isomorphisms, is closed under composition and is pullback stable.

Stable in this definition means that if $m : A \to B$ is in $\mathcal{M}$, then for arbitrary b with codomain B, the pullback

$$\begin{array}{ccc} A' & \xrightarrow{\ a\ } & A \\ {\scriptstyle m'}\downarrow & & \downarrow{\scriptstyle m} \\ B' & \xrightarrow[\ b\ ]{} & B \end{array}$$

exists and $m' \in \mathcal{M}$. A category that has a stable system of monics is referred to as an $\mathcal{M}$-category.

Lemma 1.1.14. *If $nm \in \mathcal{M}$, a stable system of monics, and m is monic, then $n \in \mathcal{M}$.*

Proof. The commutative square

$$\begin{array}{ccc} A & \xrightarrow{\ 1\ } & A \\ {\scriptstyle n}\downarrow & & \downarrow{\scriptstyle nm} \\ A' & \xrightarrow[\ m\ ]{} & B \end{array}$$

is a pullback. $\qquad\qquad\square$

Given a category $\mathbb{C}$ and a stable system of monics, the *partial map category*, $\text{Par}(\mathbb{C},\mathcal{M})$ is:

Objects: $A \in \mathbb{C}$

Equivalence Classes of Maps: $(m, f) : A \to B$ with $m : A' \to A$ is in $\mathcal{M}$ and $f : A' \to B$ is a map in $\mathbb{C}$. i.e., $\begin{array}{ccc} & A' & \\ {\scriptstyle m}\swarrow & & \searrow{\scriptstyle f} \\ A & & B \end{array}$.

Identity: $1_A, 1_A : A \to A$

Composition: via a pullback, $(m, f)(m', g) = (m''m, f'g)$ where

$$\begin{array}{ccccc} & & A'' & & \\ & {\scriptstyle m''}\swarrow & & \searrow{\scriptstyle f'} & \\ & A' & \text{(pb)} & B' & \\ {\scriptstyle m}\swarrow & & \searrow{\scriptstyle f} \quad \swarrow{\scriptstyle m'} & & \searrow{\scriptstyle g} \\ A & & B & & C \end{array}$$

Restriction: $\overline{(m, f)} = (m, m)$

For the maps, $(m, f) \sim (m', f')$ when there is an isomorphism $\gamma : A'' \to A'$ such that $\gamma m' = m$ and $\gamma f' = f$.

In [20], it is shown that:

Theorem 1.1.15 (Cockett-Lack)**.** *Every restriction category is a full subcategory of a partial map category.*

1.1.5   Restriction products and Cartesian restriction categories

Restriction categories have analogues of products and terminal objects.

Definition 1.1.16. In a restriction category $\mathbb{X}$ a *restriction product* of two objects X, Y is an object $X \times Y$ equipped with *total* projections $\pi_0 : X \times Y \to X, \pi_1 : X \times Y \to Y$ where:

$\forall f : Z \to X, g : Z \to Y, \quad \exists$ a unique $\langle f, g \rangle : Z \to X \times Y$ such that

- $\langle f, g \rangle \pi_0 \leq f$,

- $\langle f, g \rangle \pi_1 \leq g$ and

- $\overline{\langle f, g \rangle} = \overline{f}\,\overline{g} (= \overline{g}\,\overline{f})$.

Definition 1.1.17. In a restriction category $\mathbb{X}$ a *restriction terminal object* is an object $\top$ such that $\forall X$, there is a unique total map $!_X : X \to \top$ and the diagram

$$
\begin{array}{ccc}
X & \xrightarrow{\overline{f}} & X \xrightarrow{\;!_X\;} \top \\
{\scriptstyle f}\downarrow & \nearrow {\scriptstyle !_Y} & \\
Y & &
\end{array}
$$

commutes. That is, $f\,!_Y = \overline{f}\,!_X$. Note this implies that a restriction terminal object is unique up to a unique isomorphism.

Definition 1.1.18. A restriction category $\mathbb{X}$ is *Cartesian* if it has all restriction products and a restriction terminal object.

Definition 1.1.19. An object A in a Cartesian restriction category is *discrete* when the diagonal map,

$$\Delta : A \to A \times A$$

is a partial isomorphism.

A Cartesian restriction category is *discrete* when every object is discrete.

Theorem 1.1.20. *A Cartesian restriction category $\mathbb{X}$ is discrete if and only if it has meets.*

Proof. If $\mathbb{X}$ has meets, then

$$\Delta(\pi_0 \cap \pi_1) = \Delta\pi_0 \cap \Delta\pi_1 = 1 \cap 1 = 1$$

and as $\langle \pi_0, \pi_1 \rangle$ is identity,

$$\overline{\pi_0 \cap \pi_1} = \overline{\pi_0 \cap \pi_1}\langle \pi_0, \pi_1 \rangle$$

$$= \langle \overline{\pi_0 \cap \pi_1}\pi_0, \overline{\pi_0 \cap \pi_1}\pi_1 \rangle$$

$$= \langle \pi_0 \cap \pi_1, \pi_0 \cap \pi_1 \rangle$$

$$= (\pi_0 \cap \pi_1)\Delta$$

and therefore, $\pi_0 \cap \pi_1$ is $\Delta^{(-1)}$.

For the other direction, set $f \cap g = \langle f, g \rangle \Delta^{(-1)}$. By the definition of the restriction product:

$$f \cap g = \langle f, g \rangle \Delta^{(-1)} = \langle f, g \rangle \Delta^{(-1)}\Delta\pi_0 = \langle f, g \rangle \overline{\Delta^{(-1)}}\pi_0 \leq \langle f, g \rangle \pi_0 \leq f$$

Similarly, substituting π_1 for π_0 above, this gives $f \cap g \leq g$. For the left distributive law,

$$h(f \cap g) = h\langle f, g \rangle \Delta^{(-1)} = \langle hf, hg \rangle \Delta^{(-1)} = hf \cap hg$$

and finally an intersection of a map with itself is

$$f \cap f = \langle f, f \rangle \Delta^{(-1)} = (f\Delta)\Delta^{(-1)} = f\overline{\Delta} = f$$

16

as Δ is total. This shows that $\cap$ as defined above is a meet for the Cartesian restriction category $\mathbb{X}$.

$\square$

We shall refer to a Cartesian restriction category in which every object is discrete as simply a discrete restriction category.

1.1.6 Graphic Categories

In a Cartesian restriction category, a map $A \xrightarrow{f} B$ is called *graphic* when the maps

$$A \xrightarrow{\langle f,1 \rangle} B \times A \qquad \text{and} \qquad A \xrightarrow{\langle \overline{f},1 \rangle} A \times A$$

have partial inverses. A Cartesian restriction category is *graphic* when all of its maps are graphic.

Lemma 1.1.21. *In a Cartesian restriction category:*

    (i) Graphic maps are closed under composition;

    (ii) Graphic maps are closed under the restriction;

    (iii) An object is discrete if and only if its identity map is graphic.

Proof.

    (i) To show closure, it is necessary to show that $\langle fg, 1 \rangle$ has a partial inverse. By Lemma 1.1.8 on page 8, the uniqueness of the partial inverse gives

$$(\langle f,1 \rangle; \langle g,1 \rangle \times 1)^{(-1)} = \langle g,1 \rangle^{(-1)} \times 1; \langle f,1 \rangle^{(-1)}.$$

By the definition of the restriction product, $\overline{\langle fg, 1 \rangle} = \overline{fg}$. Additionally, a straightforward calculation shows that $\overline{\langle f,1 \rangle; \langle g,1 \rangle \times 1} = \overline{\langle f \langle g,1 \rangle, 1 \rangle} = \overline{f; \langle g,1 \rangle} = \overline{\langle f; g, f \rangle} = \overline{fg}\,\overline{f} = \overline{fg}$ where the last equality is by [**R.2**], [**R.3**] and finally [**R.1**].

17

Consider the diagram

$$A \xrightarrow{\langle f,1 \rangle} B \times A \xrightarrow{\langle g,1 \rangle \times 1} C \times B \times A$$

with $\langle fg,1 \rangle$ to $C \times A$ and $1 \times \langle f,1 \rangle$ from $C \times A$ to $C \times B \times A$.

From this:

$$\langle fg, 1 \rangle (1 \times \langle f, 1 \rangle)(\langle g, 1 \rangle^{(-1)} \times 1)\langle f, 1 \rangle^{(-1)} = \langle f, 1 \rangle (\langle g, 1 \rangle \times 1)(\langle g, 1 \rangle^{(-1)} \times 1)\langle f, 1 \rangle^{(-1)}$$

$$= \langle f, 1 \rangle (\overline{g \times 1})\langle f, 1 \rangle^{(-1)}$$

$$= \overline{\langle f, 1 \rangle (g \times 1)}\langle f, 1 \rangle \langle f, 1 \rangle^{(-1)}$$

$$= \overline{\langle f, 1 \rangle (g \times 1)}\langle f, 1 \rangle$$

$$= \overline{\langle f, 1 \rangle \langle f, 1 \rangle (g \times 1)}$$

$$= \overline{\langle f, 1 \rangle (g \times 1)}$$

$$= \overline{\langle fg, 1 \rangle}(= \overline{fg})$$

showing that $1 \times \langle f, 1 \rangle (\langle g, 1 \rangle^{(-1)} \times 1)\langle f, 1 \rangle^{(-1)}$ is a right inverse for $\langle fg, 1 \rangle$.

For the other direction, note that in general $hk^{(-1)} = k^{(-1)}h^{(-1)}$ and that we have $\langle fg, 1 \rangle = \langle f, 1 \rangle (\langle g, 1 \rangle \times 1)(1 \times \langle f, 1 \rangle^{(-1)})$, thus $(1 \times \langle f, 1 \rangle)(\langle g, 1 \rangle^{(-1)} \times 1)\langle f, 1 \rangle^{(-1)}$ will also be a left inverse and $\langle fg, 1 \rangle$ is a restriction isomorphism.

(ii) This follows from the definition of graphic and that $\overline{\langle f, 1 \rangle} = \overline{f} = \overline{\overline{f}}$.

(iii) Given a discrete object A, the map 1_A is graphic as $\langle 1_A, 1 \rangle = \Delta$ and therefore $\langle 1, 1 \rangle^{(-1)} = \Delta^{(-1)}$. Conversely, if $\langle 1_A, 1 \rangle$ has an inverse, then $\Delta = \langle 1_A, 1 \rangle$ has that same inverse and therefore the object is discrete.

□

Lemma 1.1.22. *A discrete restriction category is precisely a graphic Cartesian restriction category.*

Proof. The requirement is that $\langle f, 1 \rangle$ (and $\langle \overline{f}, 1 \rangle$) each have partial inverses. For $\langle f, 1 \rangle$, the inverse is $\overline{(1 \times f)\Delta^{(-1)}}\pi_1$.

To show this, calculate the two compositions. First,

$$\langle f, 1 \rangle \overline{1 \times f \Delta^{(-1)}}\pi_1 = \overline{\langle f, f \rangle \Delta^{(-1)}}\langle f, 1 \rangle \pi_1 = \overline{f \Delta \Delta^{(-1)}}\langle f, 1 \rangle \pi_1 = \overline{f}\langle f, 1 \rangle \pi_1 = \overline{f}.$$

The other direction is:

$$\overline{(1 \times f)\Delta^{(-1)}}\pi_1\langle f, 1 \rangle = \langle \overline{(1 \times f)\Delta^{(-1)}}\pi_1 f, \overline{(1 \times f)\Delta^{(-1)}}\pi_1 \rangle$$

$$= \langle \overline{(1 \times f)\Delta^{(-1)}}(1 \times f)\pi_1, \overline{(1 \times f)\Delta^{(-1)}}\pi_1 \rangle$$

$$= \langle (1 \times f)\overline{\Delta^{(-1)}}\pi_1, \overline{(1 \times f)\Delta^{(-1)}}\pi_1 \rangle$$

$$= \langle (1 \times f)\overline{\Delta^{(-1)}}\pi_0, \overline{(1 \times f)\Delta^{(-1)}}\pi_1 \rangle$$

$$= \langle \overline{(1 \times f)\Delta^{(-1)}}(1 \times f)\pi_0, \overline{(1 \times f)\Delta^{(-1)}}\pi_1 \rangle$$

$$= \langle \overline{(1 \times f)\Delta^{(-1)}}\pi_0, \overline{(1 \times f)\Delta^{(-1)}}\pi_1 \rangle$$

$$= \overline{(1 \times f)\Delta^{(-1)}}\langle \pi_0, \pi_1 \rangle$$

$$= \overline{(1 \times f)\Delta^{(-1)}}$$

The one tricky step is to realize

$$\overline{\Delta^{(-1)}}\pi_1 = \Delta^{(-1)}\Delta \pi_1$$

$$= \Delta^{(-1)}$$

$$= \Delta^{(-1)}\Delta \pi_0$$

$$= \overline{\Delta^{(-1)}}\pi_0$$

For $\langle \overline{f}, 1 \rangle$, the inverse is $\overline{(1 \times \overline{f})\Delta^{(-1)}}\pi_1$. Similarly to above,

$$\langle \overline{f}, 1 \rangle \overline{1 \times \overline{f}\Delta^{(-1)}}\pi_1 = \overline{\langle \overline{f}, \overline{f} \rangle \Delta^{(-1)}}\langle \overline{f}, 1 \rangle \pi_1 = \overline{\overline{f}\Delta \Delta^{(-1)}}\langle \overline{f}, 1 \rangle \pi_1 = \overline{\overline{f}}\langle \overline{f}, 1 \rangle \pi_1 = \overline{f}.$$

The other direction follows the same pattern as for $\langle f, 1 \rangle$. $\square$

Chapter 2

Reversible computation

Bennet, in [7], showed that it was possible to emulate a standard Turing machine via a reversible Turing machine and vice-versa. This showed the equivalence of standard and reversible Turing machines. We reproduce the essence of this proof below.

2.1 Reversible Turing machines

Turing machines consist of a tape, a read-write head positioned over the tape, a machine state and a set of instructions. The set of instructions may be given as a set of transitions determining the movement of the read-write head, what it writes and the resulting state of the machine.

Definition 2.1.1. Given an alphabet A which does not contain a space, a tape is in *standard format* when:

[***T.1***] The tape head is positioned directly over a blank space;

[***T.2***] The spaces to the left (the +1 direction) contain only elements of A.

[***T.3***] All other spaces of the tape are blank.

Definition 2.1.2. A *turing quintuple* is a quintuple $(s, \alpha, \alpha', \delta, s')$ where:

[***Q.1***] $s, s' \in S$, where S is a predefined set of states;

[***Q.2***] $\alpha, \alpha' \in A$ is predefined set of glyphs;

[***Q.3***] $\delta \in \{-1, 0, 1\}$.

Definition 2.1.3. A *standard turing quintuple set Q* consists of a set of turing quintuples such that:

(i) If $q_1 = (s_1, \alpha_1, \alpha_1', \delta_1, s_1')$ and $q_2 = (s_2, \alpha_2, \alpha_2', \delta_2, s_2')$ are in Q, then either $s_1 \neq s_2$ or $\alpha_1 \neq \alpha_2$ or both are not equal.

(ii) There are two special quintuples contained in Q:

    (a) $(s_1, \textvisiblespace, \textvisiblespace, +1, s_2)^1$, the *start quintuple*;

    (b) $(s_{t-1}, \textvisiblespace, \textvisiblespace, 0, s_t)$, the *end quintuple* where t is the number of states and is the final state of the machine.

Definition 2.1.4. A *standard Turing machine* is given by

[**TM.1**] a standard turing quintuple set;

[**TM.2**] a tape that starts in standard format;

[**TM.3**] and the condition that and if the machine halts, it will halt in state s_t, the final state of the end quintuple and the output will be in standard format.

The turing quintuples may also be regarded as giving the data for a partial function in SETS: $\tau : S \times A \to A \times \{-1, 0, 1\} \times S$.

Remark 2.1.5. A multi-tape Turing machine with n tapes and read-write heads can be described by modifying definition 2.1.4 such that α is an n-tuple of the set of glyphs for the Turing machine and δ is an n-tuple of movement directions.

Example 2.1.6. Suppose $S = \{start, run, reset, done\}$, $A = \{0, 1, \textvisiblespace\}$ and the Turing machine program is given by the quintuples

$$(start, \textvisiblespace, \textvisiblespace, +1, run),$$

$$(run, 0, 1, +1, run), (run, 1, 0, +1, run),$$

$$(run, \textvisiblespace, \textvisiblespace, -1, reset),$$

$$(reset, 0, 0, -1, reset), (reset, 1, 1, -1, reset),$$

$$(reset, \textvisiblespace, \textvisiblespace, 0, done).$$

---

[1]Here, $\textvisiblespace$ is used to signify a blank.

This program will perform a "bit-flip" of all 0s and 1s on the tape until it reads a space, reposition the read head to the standard format and then it will halt.

As we see in example , it is *possible* that a Turing machine program is reversible. If we had chosen the second quintuple to be $(run, 0, 0, +1, run)$ instead, the program would not have been reversible.

The essential property that a Turing machine program needs to be reversible is that the function τ defined from the quintuples is injective. In order to simplify the discovery the function being injective, we reformulate the turing quintuples as quadruples.

Definition 2.1.7. A *turing quadruple* is given by a quadruple

$$(s, [b_1, b_2, \ldots, b_n], [b'_1, b'_2, \ldots, b'_n], s')$$

such that:

(i) $s, s' \in S$, some set of states;

(i) $b_j \in A \cup \{\phi\}$ where A is some alphabet;

(i) $b'_j \in A + \{-1, 0, 1\}$;

(i) $b'_j \in \{-1, 0, 1\}$ if and only if $b_j = \phi$.

In this definition, $b_j = \phi$ means that the value of tape j is ignored.

A turing quadruple explicitly splits the read/write action of the Turing machine away from the movement. In a particular step for tape k, the turing machine will either read and write an item or it will not read and then move.

Remark 2.1.8. Any turing quintuple may be split into two turing quadruples by the addition of a new state a'' in A, where the first quadruple will consist of all the read-write operations and leave the Turing machine in state a''. The second quadruple will start in state a'' and all the b_j will be ϕ, with b'_j being movements on each of the n tapes.

Definition 2.1.9. A set of turing quadruples Q is called *reversible set of turing quadruples* when given $q_1, q_2 \in Q$, with $q_1 = (a, [b_j], [b'_j], a')$ and $q_2 = (c, [d_j], [d'_j], c')$:

[**RTM.1**] if $a = c$, then there is a k where $b_k, d_k \in A$ and $b_k \neq d_k$;

[**RTM.2**] if $a' = c'$, then there is a j with $b'_j, d'_j \in A$ and $b'_j \neq d'_j$.

Similarly to turing quintuples, turing quadruples may be taken as the data for a function in SETS:

$$\rho : S \times (A \cup \{\phi\}) \to (A + \{-1, 0, 1\}) \times S.$$

We can see by inspection that ρ is a reversible partial function when the set of turing quadruples that give ρ is a reversible set of turing quadruples.

Definition 2.1.10. A *reversible Turing machine* is one that is described by a set of reversible turing quadruples.

We will show that a reversible Turing machine with three tapes can emulate a Turing machine.

Theorem 2.1.11 (Bennet[7]). *Given a standard Turing Machine M, it may be emulated by a three tape reversible Turing machine R. In this case, emulated means:*

    (i) M halts on standard input I if and only if R halts on standard input $(I, \llcorner, \llcorner)$.

    (i) M halts on standard input I producint standard output O, if and only if R halts on input $(I, \llcorner, \llcorner)$ producing standard output $(I, \llcorner, O)$.

Proof. (Sketch only).

The crux of the proof is to convert the quintuples of M to the quadruples of R as noted in remark . Explicitly for a single tape machine, we have

$$(s, a, a, \delta, s') \mapsto ((s, a, a', s''), (s'', \phi, \delta, s')) . \tag{2.1}$$

In equation (2.1) on the preceding page, s'' is a new state for the machine M, not in the current set of states.

Assign an order to the n quintuples of M, where the start quintuple is the first in the order and the end quintuple comes last. Convert these to quadruples as in equation (2.1) on the previous page.

We then proceed to create three groups of quadruples for R. We call these *emulation, copy,* and *restore.*

To create the emulation phase quadruples, we examine the pairs of quadruples of M in the sorted order and produce a pair of quadruples for R.

$$\text{Pair } 1 \quad (s_1, \llcorner, \llcorner, s_1'') \mapsto (s_1, [\llcorner, \phi, \llcorner], [\llcorner, +1, \llcorner], e_1)$$

$$(s_1'', \phi, \delta, s_2) \mapsto (e_1, [\phi, \llcorner, \phi], [\delta, 1, 0], s_2)$$

$$\vdots$$

$$\text{Pair } j \quad (s_k, a_j, a_j', s_k'') \mapsto (s_k, [a_j, \phi, \llcorner], [a_j', +1, \llcorner], e_j)$$

$$(s_k'', \phi, \delta, s_i) \mapsto (e_j, [\phi, \llcorner, \phi], [\delta_j, j, 0], s_i)$$

$$\vdots$$

$$\text{Pair } n \quad (s_\ell, \llcorner, \llcorner, s_\ell'') \mapsto (s_\ell, [\llcorner, \phi, \llcorner], [\llcorner, +1, \llcorner], e_n)$$

$$(s_\ell'', \phi, 0, s_f) \mapsto (e_n, [\phi, \llcorner, \phi], [0, n, 0], s_f).$$

By inspection, one can see that even if the quadruples of M were not a reversible set, the set created for R is a reversible set, due to the writing of the quadruple index on tape 2. Upon completion of the emulation phase, tape 1 will be the same as M would have produced on its single tape, tape 2 will be $[1, 2, \ldots, n]$ and tape 3 will be blanks.

For the copy phase, we create the following quadruples:

$$(s_f, [\lrcorner, n, \lrcorner], [\lrcorner, n, \lrcorner], c_1)$$

$$(c_1, [\phi, \phi, \phi], [+1, 0, +1], c_1')$$

$$(c_1', [x, n, \lrcorner], [x, n, x], c_1) \quad \text{when } x \neq \lrcorner$$

$$(c_1', [\lrcorner, n, \lrcorner], [\lrcorner, n, x], c_2)$$

$$(c_2, [\phi, \phi, \phi], [-1, 0, -1], c_2')$$

$$(c_2', [x, n, x], [x, n, x], c_2) \quad \text{when } x \neq \lrcorner$$

$$(c_2', [\lrcorner, n_{,\rfloor}, [\lrcorner, n, \lrcorner], r_\ell).$$

In these quadruples, the states $\{c_1, c_1', c_2, c_2'\}$ should be chosen to be distinct from the states in the emulation phase. As an example, set them as follows:

$$c_1 = (\{c\}, s_1) \quad c_1' = (\{c'\}, s_1) \quad c_2 = (\{c\}, s_f) \quad c_1' = (\{c'\}, s_f).$$

At the completion of this phase, tapes 1 and 2 will be unchanged and tape 3 will be a copy of tape 1.

Finally we perform the restore phase where the history will be erased and tape 1 reset to the input. The quadruples that will accomplish this are:

$$\text{Pair } n \quad (r_n, [\phi, n, \phi], [0, \lrcorner, 0], r_n')$$

$$(r_n', [\lrcorner, \phi, \lrcorner], [\lrcorner, -1, \lrcorner], r_{n-1})$$

$$\vdots$$

$$\text{Pair } j \quad (r_k, [\phi, j, \phi], [-\delta_j, \lrcorner, 0], r_j')$$

$$(r_j', [a_j', \phi, \lrcorner], [a_j, -1, \lrcorner], r_i)$$

$$\vdots$$

$$\text{Pair } 1 \quad (r_2, [\phi, 1, \phi], [-1, \lrcorner, 0], r_1')$$

$$(r_1', [\lrcorner, \phi, \lrcorner], [\lrcorner, -1, \lrcorner], r_1).$$

The r states are derived from the s states of the emulation phase.

$$r_j = (\{r\}, s_j) \qquad r'_j = (\{r'\}, s_j).$$

In this restore phase, the indexes of the states r match up to the indexes of states s. The quadruples reverse the actions of the emulate phase on tape 1, erase the history on tape 2 and make no change to tape 3.

$\square$

2.2 Reversible automota and linear combinatory algebras

While reversible Turing machines, as described in , show that reversible computing is as powerful as standard computing, they do not give us a sense of what may be considered to be happening at a higher level.

To accomplish that task we examine the results of the paper "A Structural Approach to Reversible Computation"[1]. In this paper, Abramsky gives a description of a reversible automaton together with a linear combinatory algebra. We will begin by revisiting some definitions and constructions necessary for discussing automota. The next subsection will introduce combinatory algebras, after which we will describe the reversible automata of [1] and add a short proof that it can emulate a reversible turing machine.

2.2.1 Automota

We will describe the automota as a term-rewriting system. This requires, of course, giving a few basic definitions. See, e.g., [6].

Definition 2.2.1. An *arity* is a function from a function to the natural numbers. The arity of F is the number of inputs (arguments) required by F.

Definition 2.2.2. A *signature* Σ is a set of *function symbols* $F, G, \ldots$, each of which has an arity.

Remark 2.2.3. We refer to functions with low arity in the following ways:

- *Arity* = 0. These are known as *nullary* functions or constants.

- *Arity* = 1. These are known as *unary* functions.

- *Arity* = 2. These are known as *binary* functions.

Definition 2.2.4. A *term alphabet* is a set A containing a signature Σ and a countably infinite set X, the variables. Furthermore, $\Sigma \cap X = \phi$.

Definition 2.2.5. A *term algebra* of the term alphabet $\Sigma \cup X$ is denoted by $T_\Sigma(X)$ and defined as follows:

- $x \in V \implies x \in T_\Sigma$ and

- For any $F \in \Sigma$, with $arity(F) = n$, and $\{t_1, \ldots, t_n\} \subseteq T_\Sigma$, then $F(t_1, \ldots, t_n) \in T_\Sigma$. In the case where $arity(F) = 0$, we write $F \in T_\Sigma$.

Definition 2.2.6. The *ground terms* of a term algebra are those terms that do not contain any variable. The set of these terms is designated as T_Σ.

Remark 2.2.7. Note the ground terms consist of the constants and recursively applying the function symbols of Σ to them.

As we are considering rewrite systems, we will need to consider aspects of substitution and unification.

Definition 2.2.8. A *substitution* is a map $\sigma : T_\Sigma(X) \to T_\Sigma(X)$ which is natural for all function symbols in Σ. In particular if $arity(c) = 0$ then $\sigma(c) = c$.

Note that given the above definition a substitution σ is completely determined by its action on variables. If $\sigma : X \to X$ and is injective, we call σ a renaming. Moreover, if σ restricted to the variables in a term t is an injective map of X on those variables, we call *sigma* a renaming of t.

Substitution allows us to define a partial order on $T_\Sigma(X)$, as follows:

Definition 2.2.9. In $T_\Sigma(X)$, let $\sigma(t) = s$. Then we say s is an *instance* of t, written $s \preceq t$. Moreover, if σ is not just a renaming for t, then we write $s \prec t$. If σ is a renaming of t, we write $s \simeq t$.

Lemma 2.2.10. *Subsumption, as defined in 2.2.9 is a partial order, i.e., it is transitive and reflexive.*

Proof.   □

Lemma 2.2.11. *Given terms r, t such that there is at least one s with $s \preceq r$ and $s \preceq t$, then there exists a g such that $g \preceq r$ and $g \preceq t$ and for any s' with $s' \preceq r$ and $s' \preceq t$ we will have $s' \preceq g$.*

Proof.

1. Algorithm to compute supremum of p, q terms.

2. Strict $\prec$ has no infinite ascending chains.

3. Shows main part - there exists.

4. Can now show it is unique up to renaming.

                                                                                    □

The subsumption ordering can be used to derive a similar ordering on substitutions:

Definition 2.2.12. $\sigma \preceq \tau$ if and only if there is a ρ with $\sigma = \tau\rho$, where $\tau\rho$ is the diagrammatic order composition of the two substitutions.

Definition 2.2.13. For terms s, t, if $\sigma(t) = \sigma(s)$, then the substitution σ is called a *unifier* for the terms s, t.

Lemma 2.2.14. *If s, t are terms with a unifier σ, there exists a substitution τ that unifies s, t such that $\tau \preceq \rho$ whenever ρ unifies s, t. ρ is called the* most general unifier *of s and t.*

Proof. Follows from 2.2.11 on the preceding page. $\square$

Notation 2.2.15. Following [1], we write $\mathcal{U}(t, u) \downarrow \sigma$ if σ is the most general unifier of terms t, u.

2.2.2 Combinatory Algebra

Definition 2.2.16. A *combinatory algebra* is an algebra with one binary operation, $\cdot$ written in infix notation. The operation is not assumed to be associative. Multi-element expressions such as $a \cdot b \cdot c$ are to be taken as associating to the left, that is,

$$a \cdot b \cdot c = (a \cdot b) \cdot c.$$

The combinatory algebra may possess distinguished elements that are subject to specific rewrite rules.

Definition 2.2.17. *Combinatory logic* is the combinatory algebra with two distinguished elements, K and S, such that the following hold:

$$
\begin{aligned}
K \cdot x \cdot y &= x \\
S \cdot x \cdot y \cdot z &= x \cdot z \cdot (y \cdot z).
\end{aligned}
$$

Note that combinatory logic does not require a specific set that must be used for the algebra, simply that it has the two distinguished elements.

Combinatory logic was shown to be equivalent to the λ calculus by

For example, we may define the identity combinator I as $I = S \cdot K \cdot K$. Further combinators may be defined, such as the B combinator, defined by $B \cdot a \cdot b \cdot c = a \cdot (b \cdot c)$. The S and K combinators are complete, in that other combinators such as B may be defined from them. E.g., $B = S \cdot (K \cdot S) \cdot K$. In fact, we may define an alternate combinatory algebra that is equivalent to Combinatory Logic.

Definition 2.2.18. A *BCKW-Combinatory algebra* is a Combinatory Algebra with four distinguish elements, B, C, K, and W subject to the following equations:

$$
\begin{aligned}
B \cdot a \cdot b \cdot c &= a \cdot (b \cdot c) \\
C \cdot a \cdot b \cdot c &= a \cdot c \cdot b \\
K \cdot a \cdot b &= a \\
W \cdot a \cdot b &= a \cdot b \cdot b
\end{aligned}
$$

In fact, a BCKW-Combinatory algebra is equivalent to a Combinatory logic.

Lemma 2.2.19. *The distinguished elements of a BCKW-Combinatory algebra may be represented by S and K. Conversly, the S and K of a Combinatory logic may be created from B,C,K and W.*

Proof. For the first statement, we have:

$$
\begin{aligned}
B &= S \cdot (K \cdot S) \cdot K \\
C &= S \cdot (S \cdot (K \cdot (S \cdot (K \cdot S) \cdot K)) \cdot S) \cdot (K \cdot K) \\
K &= K \\
W &= S \cdot S \cdot (S \cdot K).
\end{aligned}
$$

Going the other direction, we have:

$$
\begin{aligned}
I &= W \cdot K \\
K &= K \\
S &= B \cdot (B \cdot (B \cdot W) \cdot C) \cdot (B \cdot B) \text{ and} \\
&= B \cdot (B \cdot W) \cdot (B \cdot B \cdot C).
\end{aligned}
$$

We show the computations of B and S in detail.

$$
\begin{aligned}
B \cdot a \cdot b \cdot c &= S \cdot (K \cdot S) \cdot K \cdot a \cdot b \cdot c \\
&= (K \cdot S) \cdot a \cdot (K \cdot a) \cdot b \cdot c \\
&= S \cdot (K \cdot a) \cdot b \cdot c \\
&= K \cdot a \cdot c \cdot (b \cdot c) \\
&= a \cdot (b \cdot c)
\end{aligned}
$$

$$
\begin{aligned}
S \cdot a \cdot b \cdot c &= B \cdot (B \cdot W) \cdot (B \cdot B \cdot C) \cdot a \cdot b \cdot c \\
&= (B \cdot W) \cdot ((B \cdot B \cdot C) \cdot a) \cdot b \cdot c \\
&= B \cdot W \cdot ((B \cdot B \cdot C) \cdot a) \cdot b \cdot c \\
&= W \cdot (((B \cdot B \cdot C) \cdot a) \cdot b) \cdot c \\
&= (((B \cdot B \cdot C) \cdot a) \cdot b) \cdot c \cdot c \\
&= B \cdot B \cdot C \cdot a \cdot b \cdot c \cdot c \\
&= B \cdot (C \cdot a) \cdot b \cdot c \cdot c \\
&= (C \cdot a) \cdot (b \cdot c) \cdot c \\
&= C \cdot a \cdot (b \cdot c) \cdot c \\
&= a \cdot c \cdot (b \cdot c)
\end{aligned}
$$

$\square$

If we use the notation $a^n \cdot b$ to mean $a \cdot a \cdots \cdots a \cdot b$ where a is repeated n times, then we can terms which correspond to the Church numbers of lambda calculus:

$$\bar{n} \equiv (\mathrm{S} \cdot \mathrm{B})^n \cdot (\mathrm{K} \cdot \mathrm{I})$$

Definition 2.2.20. A partial function $f : \mathbb{N} \to \mathbb{N}$ is *representable* in combinatory logic if there is a term M_f such that $M_f \cdot \bar{n} = \bar{m}$ whenever $f(n) = m$ and $M_f \cdot \bar{n}$ does not have a normal form if $f(n) \uparrow$.

When we say that combinatory logic with S and K is complete, we mean the following theorem:

Theorem 2.2.21. *The partial functions that are representable in combinatory logic are exactly the partial recursive functions.*

2.2.3 Linear Combinatory Algebra

Definition 2.2.22. A *Linear Combinatory Algebra* $(A, \cdot, !)$ is an algebra A with an applicative binary operation $\cdot$, an unary operator $! : A \to A$ and eight distinquished elements: B, C, I, K, D, δ, F and W in A which satisfy the following rules:

1. $\mathrm{B} \cdot a \cdot b \cdot c \qquad = a \cdot (b \cdot c)$
2. $\mathrm{C} \cdot a \cdot b \cdot c \qquad = a \cdot c \cdot b$
3. $\mathrm{I} \cdot a \qquad\qquad = a$
4. $\mathrm{K} \cdot a \cdot !b \qquad\quad = a$
5. $\mathrm{D} \cdot !a \qquad\qquad = a$
6. $\delta \cdot !a \qquad\qquad = !!a$
7. $\mathrm{F} \cdot !a \cdot !b \qquad\quad = !(a \cdot b)$
8. $\mathrm{W} \cdot a \cdot !b \qquad\quad = a \cdot !b \cdot !b$

Note that a Linear Combinatory Algebra always contains a BCKW-Combinatory algebra.

Define $\mathrm{D}' = \mathrm{C} \cdot (\mathrm{B} \cdot \mathrm{B} \cdot \mathrm{I}) \cdot (\mathrm{B} \cdot \mathrm{D} \cdot \mathrm{I})$ and the binary operator $\bullet$ on A such that $a \bullet b \equiv a \cdot !b$.

Then, define the following:

$$B_s = C \cdot (B \cdot (B \cdot B \cdot B) \cdot (D' \cdot I)) \cdot (C \cdot ((B \cdot B) \cdot F) \cdot \delta)$$

$$C_s = D' \cdot C$$

$$K_s = D' \cdot K$$

$$W_s = D' \cdot W.$$

Lemma 2.2.23. *Given and Linear Combinatory Algebra* $(A, \cdot, !)$, *then* $(A, \bullet)$ *is a BCKW-Combinatory algebra with* B, C, K, W *set to* B_s, C_s, K_s, W_s *from above.*

Proof. We show the calculation for K_s, the others are similar.

$$
\begin{aligned}
K_s \bullet a \bullet b &\equiv D' \cdot K \cdot !a \cdot !b \\
&= C \cdot (B \cdot B \cdot I) \cdot (B \cdot D \cdot I) \cdot K \cdot !a \cdot !b \\
&= (B \cdot B \cdot I \cdot K) \cdot (B \cdot D \cdot I) \cdot !a \cdot !b \\
&= B \cdot (I \cdot K) \cdot (B \cdot D \cdot I) \cdot !a \cdot !b \\
&= (I \cdot K) \cdot ((B \cdot D \cdot I) \cdot !a) \cdot !b \\
&= K \cdot ((B \cdot D \cdot I) \cdot !a) \cdot !b \\
&= (B \cdot D \cdot I) \cdot !a \\
&= D \cdot (I \cdot !a) \\
&= D \cdot !a \\
&= a
\end{aligned}
$$

□

Chapter 3

Inverse categories

3.1 Inverse products

Our goal is now to add "products", to an inverse category. Because an inverse category that has a restriction product is a restriction preorder, what is meant by "product" must be specialized for the inverse setting. These we call *inverse products*, which are defined in sub-section below.

Inverse products are given by a tensor product which supports a diagonal, but lack projections. The diagonal map is required to give a natural Frobenius structure to each object.

3.1.1 Inverse categories with restriction products

We start by showing than an inverse category with restriction products is a restriction preorder. Thus simply using restriction products provides a notion which is too narrow.

Definition 3.1.1. Two parallel maps $f, g : A \to B$ in a restriction category are *compatible*, written as $f \smile g$, when $\overline{f}g = \overline{g}f$.

Definition 3.1.2. A restriction category $\mathbb{X}$ is a *restriction preorder* when all parallel pairs of maps are compatible.

Lemma 3.1.3. *Given an inverse category $\mathbb{X}$, if it has restriction products, it is a restriction preorder. That is,*

$$A \underset{g}{\overset{f}{\rightrightarrows}} B \implies f \smile g.$$

Proof. Notice,

$$\pi_1{}^{(-1)} = \Delta\pi_1\pi_1{}^{(-1)}$$

$$= \Delta\overline{\pi_1}$$

$$= \Delta.$$

This gives $\overline{\pi_1{}^{(-1)}} = 1$ and therefore π_1 (and similarly, π_0) is an isomorphism.

Starting with the product map $\langle f, g \rangle$,

$$\frac{\frac{\frac{\frac{\langle f,g \rangle = \langle f,g \rangle}{\langle f,g \rangle \pi_1 \pi_1{}^{(-1)} = \langle f,g \rangle \pi_0 \pi_0{}^{(-1)}}}{\overline{f}g\pi_1{}^{(-1)} = \overline{g}f\pi_0{}^{(-1)}}}{\overline{f}g\Delta = \overline{g}f\Delta}}{\overline{f}g = \overline{g}f}$$

which shows that f and g are compatible. $\qquad\square$

Corollary 3.1.4. $\mathbb{X}$ *is an Cartesian inverse category if and only if Total($K_r(\mathbb{X})$) is a meet preorder.*

Proof. Total($\mathbb{X}$), the subcategory of total maps on $\mathbb{X}$, has products and therefore every pair of parallel maps is compatible. However, total compatible maps are simply equal, therefore there is at most one map between any two objects. Hence, it is a preorder with the meet being the product.

Similarly, from [19] and [21], Total($K_r(\mathbb{X})$) is an inverse category and has products and is therefore also a meet preorder.

When Total($K_r(\mathbb{X})$) is a meet preorder, define the product as the meet of the maps and the terminal object as the supremum of all maps. $\qquad\square$

Corollary 3.1.5. *Every Cartesian inverse category is a full subcategory of a partial map category of a meet semi-lattice.*

3.1.2 Inverse products

An *inverse product* on a restriction category $\mathbb{X}$ is given by a tensor $\otimes$ together with a natural "Frobenius" diagonal map, Δ. The data for the tensor is:

$$\_ \otimes \_ : \mathbb{X} \times \mathbb{X} \to \mathbb{X} \quad \text{(a restriction functor)}$$

$$1 : \mathbf{1} \to \mathbb{X}$$

$$u_\otimes^l : 1 \otimes A \to A$$

$$u_\otimes^r : A \otimes 1 \to A$$

$$a_\otimes : (A \otimes B) \otimes C \to A \otimes (B \otimes C)$$

$$c_\otimes : A \otimes B \to B \otimes A$$

where $u_\otimes^l, u_\otimes^r, a_\otimes, c_\otimes$ are all natural isomorphisms and the standard symmetric monoidal equations and coherence diagrams hold (see, e.g., [14]). Note that as all the coherence maps are isomorphisms, they are total. Additionally, we define the map $ex_\otimes : (A \otimes B) \otimes (C \otimes D) \to (A \otimes C) \otimes (B \otimes D)$

$$ex_\otimes = a_\otimes (1 \otimes a_\otimes^{(-1)})(1 \otimes (c_\otimes \otimes 1))(1 \otimes a_\otimes)a_\otimes^{(-1)}.$$

The diagonal map $\Delta_A : A \to A \otimes A$ must be total and must satisfy the following:

Co-commutative

$$
\begin{array}{ccc}
A & \xrightarrow{\quad\Delta\quad} & A \otimes A \\
\Delta \downarrow & & \downarrow 1\otimes\Delta \\
A \otimes A & & A \otimes (A \otimes A) \\
\searrow_{\Delta\otimes 1} & & \nearrow_{a_\otimes} \\
& (A \otimes A) \otimes A &
\end{array}
$$

Co-associative

$$
\begin{array}{ccc}
A \otimes B & \xrightarrow{\quad\Delta\otimes\Delta\quad} & (A \otimes A) \otimes (B \otimes B) \\
\Delta \downarrow & & \downarrow ex_\otimes \\
(A \otimes B) \otimes (A \otimes B) & =\!\!=\!\!=\!\!=\!\!= & (A \otimes B) \otimes (A \otimes B)
\end{array}
$$

Exchange

$$
\begin{array}{ccc}
A \otimes A & \xrightarrow{\quad(\Delta\otimes 1)a_\otimes\quad} & A \otimes (A \otimes A) \\
{\scriptstyle(1\otimes\Delta)a_\otimes{}^{(-1)}}\downarrow \quad \searrow^{\Delta^{(-1)}} & & \downarrow {\scriptstyle 1\otimes\Delta^{(-1)}} \\
 & A \searrow_{\Delta} & \\
(A \otimes A) \otimes A & \xrightarrow{\quad\Delta^{(-1)}\otimes 1\quad} & A \otimes A
\end{array}
$$

Frobenius

Thus, Δ is a co-commutative, coassociative map which together with $\Delta^{(-1)}$ forms a Frobenius algebra.

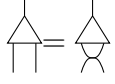**Remark 3.1.6.** Note also, co-commutativity implies that $c_\otimes \Delta^{(-1)} = \Delta^{(-1)}$. One can see this as:

$$
\Delta(c_\otimes \Delta^{(-1)}) = (\Delta c_\otimes)\Delta^{(-1)} = \Delta\Delta^{(-1)} = \overline{\Delta} \text{ and}
$$
$$
(c_\otimes \Delta^{(-1)})\Delta = (c_\otimes \Delta^{(-1)})(\Delta c_\otimes) = \overline{c_\otimes \Delta^{(-1)}}.
$$

But this means that both $\Delta^{(-1)}$ and $c_\otimes \Delta^{(-1)}$ are partial inverses for Δ and are therefore equal.

Similarly, one can show that $(\Delta^{(-1)} \otimes 1)\Delta^{(-1)} = a_\otimes (\Delta^{(-1)} \otimes 1)\Delta^{(-1)}$.

Diagrammatic language

Inverse products are extra structure on an inverse category, rather than a property. A concrete category showing this is given in the following example.

Example 3.1.7 (Showing that inverse product is additional structure.)**.**

Any discrete category (i.e., a category with only the identity arrows) is a trivial inverse category. To create an inverse product on the category, add a commutative, associative, idempotent multiplication, with a unit, on the objects.

Label the four objects of $\mathbb{D}$ as a, b, c and d. Then, define two different inverse product tensors by:

$\otimes$	a	b	c	d
a	a	a	a	a
b	a	b	**b**	b
c	a	**b**	c	c
d	a	b	c	d

$\otimes$	a	b	c	d
a	a	a	a	a
b	a	b	**a**	b
c	a	**a**	c	c
d	a	b	c	d

The fact that these operations are idempotent(commutative and associative) implies there is a trivial Frobenius structure.

3.1.3   Discrete inverse categories

An inverse category with inverse products is a *discrete inverse category*. This paper will now present some properties of discrete inverse categories. These properties will be used later when describing a functor that lifts the inverse category to a Cartesian restriction category.

Lemma 3.1.8. *In a discrete inverse category* $\mathbb{X}$ *with the tensor* $\otimes$ *and* Δ *defined as above, where* $e = \bar{e}$ *is a restriction idempotent and* f, g, h *are arrows in* $\mathbb{X}$*, the following are true:*

(i) $e = \Delta(e \otimes 1)\Delta^{(-1)}$.

(ii) $e\Delta(f \otimes g) = \Delta(ef \otimes g)$ *(and* $= \Delta(f \otimes eg)$ *and* $= \Delta(ef \otimes eg)$*.)*

(iii) $(f \otimes ge)\Delta^{(-1)} = (f \otimes g)\Delta^{(-1)}e$ *(and* $= (fe \otimes g)\Delta^{(-1)}$ *and* $= (fe \otimes ge)\Delta^{(-1)}$*.)*

(iv) $\overline{\Delta(f \otimes g)\Delta^{(-1)}} = \Delta(1 \otimes gf^{(-1)})\Delta^{(-1)}$.

(v) *If* $\Delta(h \otimes g)\Delta^{(-1)} = \overline{\Delta(h \otimes g)\Delta^{(-1)}}$ *then* $(\Delta(h \otimes g)\Delta^{(-1)})h = \Delta(h \otimes g)\Delta^{(-1)}$.

(vi) $\Delta(f \otimes 1) = \Delta(g \otimes 1) \implies f = g$.

(vii) $(f \otimes 1) = (g \otimes 1) \implies f = g$.

Proof.

(1)This is shown by proving both sides equal $\Delta(e \otimes 1)\Delta^{(-1)}\Delta(e \otimes 1)\Delta^{(-1)}$.

$$\Delta(e \otimes 1)\Delta^{(-1)}\Delta(e \otimes 1)\Delta^{(-1)} = \Delta(e \otimes 1)\Delta^{(-1)}\Delta(1 \otimes e)\Delta^{(-1)} \quad \text{cocommutativity}$$

$$= \Delta(e\Delta \otimes 1)(1 \otimes \Delta^{(-1)}e)\Delta^{(-1)} \quad \text{Frobenius}$$

$$= \Delta(\Delta \otimes 1)(e \otimes e \otimes 1)(1 \otimes \Delta^{(-1)}e)\Delta^{(-1)} \quad \Delta\text{natural}$$

$$= \Delta(\Delta \otimes 1)(e \otimes e \otimes 1)(1 \otimes e \otimes e)(1 \otimes \Delta^{(-1)})\Delta^{(-1)} \quad \Delta^{(-1)} \text{ natural}$$

$$= \Delta(\Delta \otimes 1)(e \otimes e \otimes e)(1 \otimes \Delta^{(-1)})\Delta^{(-1)} \quad e \text{ idempotent}$$

$$= \Delta(\Delta \otimes 1)(e \otimes \Delta^{(-1)}e)\Delta^{(-1)} \quad \Delta^{(-1)} \text{ natural}$$

$$= \Delta(\Delta \otimes 1)(1 \otimes \Delta^{(-1)})\Delta^{(-1)}e \quad \Delta^{(-1)} \text{ natural}$$

$$= \Delta\Delta^{(-1)}\Delta\Delta^{(-1)}e \quad \text{Frobenius}$$

$$= e \quad \Delta \text{ total.}$$

At the same time,

$$\Delta(e \otimes 1)\Delta^{(-1)}\Delta(e \otimes 1)\Delta^{(-1)} = \Delta(e\Delta \otimes 1)(e \otimes \Delta^{(-1)}1)\Delta^{(-1)} \qquad \text{Frobenius}$$

$$= \Delta(\Delta \otimes 1)(e \otimes e \otimes 1)(e \otimes \Delta^{(-1)})\Delta^{(-1)} \qquad \Delta\text{natural}$$

$$= \Delta(\Delta \otimes 1)(e \otimes e \otimes 1)(e \otimes 1 \otimes 1)(1 \otimes \Delta^{(-1)})\Delta^{(-1)} \quad \Delta^{(-1)} \text{ natural}$$

$$= \Delta(\Delta \otimes 1)(e \otimes e \otimes 1)(1 \otimes \Delta^{(-1)})\Delta^{(-1)} \qquad e \text{ idempotent}$$

$$= \Delta(e\Delta \otimes 1)(1 \otimes \Delta^{(-1)})\Delta^{(-1)} \qquad \Delta\text{natural}$$

$$= \Delta(e \otimes 1)\Delta^{(-1)}\Delta\Delta^{(-1)} \qquad \text{Frobenius}$$

$$= \Delta(e \otimes 1)\Delta^{(-1)} \qquad \Delta \text{ total}$$

which gives $e = \Delta(e \otimes 1)\Delta^{(-1)}$.

(This equality starts by using the previous equality:

$$e\Delta(f \otimes g) = \Delta(e \otimes 1)\Delta^{(-1)}\Delta(f \otimes g) \qquad \text{by part (i)}$$

$$= \Delta(e \otimes 1)\overline{\Delta^{(-1)}}(f \otimes g)$$

$$= \Delta\overline{\Delta^{(-1)}}(e \otimes 1)(f \otimes g) \qquad [\boldsymbol{R.2}] \text{ as } e \otimes 1 \text{ is a restriction idempotent}$$

$$= \Delta(ef \otimes g) \qquad (ff^{(-1)} = f).$$

The second and third equalities follow by cocommutativity, naturality of Δ and e being a restriction idempotent.

(As in the preceding page), details are only given for the first equality.

$$(f \otimes g)\Delta^{(-1)}e$$

$$= (f \otimes g)\Delta^{(-1)}\Delta(1 \otimes e)\Delta^{(-1)} \qquad \text{part (i)}$$

$$= (f \otimes g)\overline{\Delta^{(-1)}}(1 \otimes e)\Delta^{(-1)}$$

$$= (f \otimes g)(1 \otimes e)\overline{\Delta^{(-1)}}\Delta^{(-1)} \qquad [\boldsymbol{R.2}]$$

$$= (f \otimes ge)\Delta^{(-1)} \qquad [\boldsymbol{R.1}]$$

The other equalities follow from co-commutativity, naturality of Δ and e being a restriction idempotent.

(Here we start by using the fact all maps have a partial inverse:

$$\overline{\Delta(f \otimes g)\Delta^{(-1)}}$$

$$= \Delta(f \otimes g)\Delta^{(-1)}\Delta(f^{(-1)} \otimes g^{(-1)})\Delta^{(-1)}$$

$$= \Delta(g \otimes f)\Delta^{(-1)}\Delta(g^{(-1)} \otimes f^{(-1)})\Delta^{(-1)} \qquad\qquad \text{co-commutative}$$

$$= \Delta(g\Delta \otimes f)(g^{(-1)} \otimes \Delta^{(-1)}f^{(-1)})\Delta^{(-1)} \qquad\qquad \text{Frobenius}$$

$$= \Delta(\Delta \otimes 1)(g \otimes g \otimes f)(g^{(-1)} \otimes \Delta^{(-1)}f^{(-1)})\Delta^{(-1)} \qquad\qquad \Delta \text{ natural}$$

$$= \Delta(\Delta \otimes 1)(g \otimes g \otimes f)(g^{(-1)} \otimes f^{(-1)} \otimes f^{(-1)})(1 \otimes \Delta^{(-1)})\Delta^{(-1)} \qquad\qquad \Delta^{(-1)} \text{ natural}$$

$$= \Delta(\Delta \otimes 1)(\overline{g} \otimes gf^{(-1)} \otimes \overline{f})(1 \otimes \Delta^{(-1)})\Delta^{(-1)} \qquad\qquad \text{combine maps}$$

$$= \Delta(\Delta \otimes 1)(\overline{g} \otimes \overline{g}\,gf^{(-1)}\overline{f} \otimes \overline{f})(1 \otimes \Delta^{(-1)})\Delta^{(-1)} \qquad\qquad \text{restriction axioms}$$

$$= \Delta(\overline{g}\Delta \otimes 1)(1 \otimes gf^{(-1)}\overline{f} \otimes \overline{f})(1 \otimes \Delta^{(-1)})\Delta^{(-1)} \qquad\qquad \Delta \text{ natural}$$

$$= \Delta(\overline{g}\Delta \otimes 1)(1 \otimes gf^{(-1)} \otimes 1)(1 \otimes \Delta^{(-1)}\overline{f})\Delta^{(-1)} \qquad\qquad \Delta^{(-1)} \text{ natural}$$

$$= \Delta(\Delta \otimes 1)(1 \otimes \overline{g}gf^{(-1)} \otimes 1)(1 \otimes \Delta^{(-1)}\overline{f})\Delta^{(-1)} \qquad\qquad \text{This lemma((ii))}$$

$$= \Delta(\Delta \otimes 1)(1 \otimes \overline{g}gf^{(-1)}\overline{f} \otimes 1)(1 \otimes \Delta^{(-1)})\Delta^{(-1)} \qquad\qquad \text{This lemma((iii))}$$

$$= \Delta(\Delta \otimes 1)(1 \otimes gf^{(-1)} \otimes 1)(1 \otimes \Delta^{(-1)})\Delta^{(-1)} \qquad\qquad \text{restriction axioms}$$

$$= \Delta c_{A,A}(\Delta \otimes 1)(1 \otimes gf^{(-1)} \otimes 1)(1 \otimes \Delta^{(-1)})\Delta^{(-1)} \qquad\qquad \text{co-commutative}$$

$$= \Delta(1 \otimes \Delta)c_{A,A\otimes A}(1 \otimes gf^{(-1)} \otimes 1)(1 \otimes \Delta^{(-1)})\Delta^{(-1)} \qquad\qquad c_{\otimes}\text{natural}$$

$$= \Delta(1 \otimes \Delta)(1 \otimes 1 \otimes gf^{(-1)})c_{A,A\otimes A}(1 \otimes \Delta^{(-1)})\Delta^{(-1)} \qquad\qquad c_{\otimes}\text{natural}$$

$$= \Delta(1 \otimes \Delta)(1 \otimes 1 \otimes gf^{(-1)})(\Delta^{(-1)} \otimes 1)c_{A,A}\Delta^{(-1)} \qquad\qquad c_{\otimes}\text{natural}$$

$$= \Delta(1 \otimes \Delta)(1 \otimes 1 \otimes gf^{(-1)})(\Delta^{(-1)} \otimes 1)\Delta^{(-1)} \qquad\qquad c_{\otimes}\text{co-commutative}$$

$$= \Delta\Delta^{(-1)}\Delta(1 \otimes gf^{(-1)})\Delta^{(-1)} \qquad\qquad \text{Frobenius}$$

$$= \Delta(1 \otimes gf^{(-1)})\Delta^{(-1)} \qquad\qquad \Delta \text{ total}$$

Note the pattern in the last few lines of using the co-commutativity of Δ,

the naturality of the commutativity isomorphism and finishing with the co-commutativity of $\Delta^{(-1)}$. In future proofs, these steps will be combined to a single line and referred to as commutativity.

(Beginning page 36 with the assumption,

$$(\Delta(h \otimes g)\Delta^{(-1)})h = \overline{\Delta(h \otimes g)\Delta^{(-1)}}h$$

$$= \Delta(1 \otimes gh^{(-1)})\Delta^{(-1)}h \qquad \text{This lemma(\textcolor{red}{(iv)})}$$

$$= \Delta(1 \otimes gh^{(-1)})\Delta^{(-1)}\Delta(h \otimes h)\Delta^{(-1)} \qquad \Delta \text{ total and natural}$$

$$= \Delta(1 \otimes gh^{(-1)})(\Delta \otimes 1)(1 \otimes \Delta^{(-1)})(h \otimes h)\Delta^{(-1)} \qquad \text{Frobenius}$$

$$= \Delta(\Delta \otimes 1)(1 \otimes 1 \otimes gh^{(-1)})(1 \otimes \Delta^{(-1)})(h \otimes h)\Delta^{(-1)} \qquad \Delta \text{ natural}$$

$$= \Delta(\Delta \otimes 1)(1 \otimes 1 \otimes gh^{(-1)})(h \otimes h \otimes h)(1 \otimes \Delta^{(-1)})\Delta^{(-1)} \qquad \Delta^{(-1)} \text{ natural}$$

$$= \Delta(\Delta \otimes 1)(h \otimes h \otimes gh^{(-1)}h)(1 \otimes \Delta^{(-1)})\Delta^{(-1)} \qquad \text{combine terms}$$

$$= \Delta(h \otimes g\overline{h^{(-1)}})(\Delta \otimes 1)(1 \otimes \Delta^{(-1)})\Delta^{(-1)} \qquad \Delta \text{ natural}$$

$$= \Delta(h \otimes g\overline{h^{(-1)}})\Delta^{(-1)}\Delta\Delta^{(-1)} \qquad \text{Frobenius}$$

$$= \Delta(h \otimes g\overline{h^{(-1)}})\Delta^{(-1)} \qquad \Delta \text{ total}$$

$$= \Delta(h \otimes g)\Delta^{(-1)}\overline{h^{(-1)}} \qquad \text{part (\textcolor{red}{(ii)})}$$

$$= \Delta(h\overline{h^{(-1)}} \otimes g)\Delta^{(-1)} \qquad \text{part (\textcolor{red}{(ii)})}$$

$$= \Delta(h \otimes g)\Delta^{(-1)} \qquad \text{property of inverse.}$$

and natural, we start with:

$$f = \Delta(f \otimes f)\Delta^{(-1)}$$
$$= \Delta(f \otimes 1)(1 \otimes f)\Delta^{(-1)}$$
$$= \Delta(g \otimes 1)(1 \otimes f)\Delta^{(-1)} \qquad\qquad \text{assumption}$$
$$= \Delta(1 \otimes f)(g \otimes 1)\Delta^{(-1)} \qquad\qquad \text{Identities commute}$$
$$= \Delta(1 \otimes g)(g \otimes 1)\Delta^{(-1)} \qquad\qquad \text{assumption, co-commutative}$$
$$= \Delta(g \otimes g)\Delta^{(-1)}$$
$$= g\Delta\Delta^{(-1)} \qquad\qquad \Delta \text{ natural}$$
$$= g \qquad\qquad \Delta \text{ total.}$$

$\square$

Proposition 3.1.9. *A discrete inverse category has meets, where* $f \cap g = \Delta(f \otimes g)\Delta^{(-1)}$.

Proof. $f \cap g \leq f$:

$$f \cap g = \Delta(f \otimes g)\Delta^{(-1)} \qquad\qquad \text{Definition of } \cap$$
$$= \Delta(f\overline{f^{(-1)}} \otimes g)\Delta^{(-1)} \qquad\qquad \text{property of inverse}$$
$$= \Delta(f \otimes g\overline{f^{(-1)}})\Delta^{(-1)} \qquad\qquad \text{by lemma 3.1.8((iii))}$$
$$= \Delta(f \otimes gf^{(-1)}f)\Delta^{(-1)} \qquad\qquad \text{definition of inverse}$$
$$= \Delta(1 \otimes gf^{(-1)})\Delta^{(-1)}f \qquad\qquad \Delta^{(-1)} \text{ natural}$$
$$= \overline{f \cap g}f \qquad\qquad \text{by lemma 3.1.8((iv))}$$

$f \cap f = f$:

$$f \cap f = \Delta(f \otimes f)\Delta^{(-1)}$$

$$= f\Delta\Delta^{(-1)} \qquad\qquad \Delta \text{ natural}$$

$$= f \qquad\qquad \Delta \text{ total.}$$

$h(f \cap g) = hf \cap hg$:

$$h(f \cap g) = h\Delta(f \otimes g)\Delta^{(-1)} \qquad\qquad \text{Definition of } \cap$$

$$= \Delta(h \otimes h)(f \otimes g)\Delta^{(-1)} \qquad\qquad \Delta \text{ natural}$$

$$= \Delta(hf \otimes hg)\Delta^{(-1)} \qquad\qquad \text{compose maps}$$

$$= hf \cap hg \qquad\qquad \text{Definition of } \cap.$$

□

3.1.4 The inverse subcategory of a discrete restriction category

Given a discrete restriction category, one can pick out the maps which are partial isomorphisms. Using results from the previous sub-section and from sub-section sub-section 1.1.6 on page 17, this section will show that these maps form a restriction subcategory and in fact, form a discrete inverse category.

Lemma 3.1.10. *Given $\mathbb{X}$ is a discrete restriction category, the invertible maps of $\mathbb{X}$, together with the objects of $\mathbb{X}$ form a sub restriction category which is a discrete inverse category, denoted by $Inv(\mathbb{X})$.*

Proof. As shown in Lemma 1.1.8 on page 8, partial isomorphisms are closed under composition. The identity maps are in $Inv(\mathbb{X})$. Trivially, restrictions of partial isomorphisms are also partial isomorphisms.

The product on the discrete restriction category $\mathbb{X}$ becomes the tensor product of the restriction category $Inv(\mathbb{X})$. Table table 3.1 on the next page shows how each of the elements

of the tensor are defined. Note that the last definition makes explicit use of the fact we are in a discrete restriction category and hence the Δ of $\mathbb{X}$ possesses a partial inverse.

$\mathbb{X}$	$Inv(\mathbb{X})$	Inverse map
$A \times B$	$A \otimes B$	
$\top$	1	
$\pi_1 : \top \times A \to A$	$u_\otimes^l : 1 \otimes A \to A$	$\langle !, 1 \rangle$
$\pi_0 : A \times \top \to A$	$u_\otimes^r : A \otimes 1 \to A$	$\langle 1, ! \rangle$
$\langle \pi_0 \pi_0, \langle \pi_0 \pi_1, \pi_1 \rangle \rangle : (A \times B) \times C \to A \times (B \times C)$	$a_\otimes : (A \otimes B) \otimes C \to A \otimes (B \otimes C)$	$\langle \langle \pi_0, \pi_1 \pi_0 \rangle, \pi_1 \pi_1 \rangle$
$\langle \pi_1, \pi_0 \rangle : A \times B \to B \times A$	$c_\otimes : A \otimes B \to B \otimes A$	$\langle \pi_1, \pi_0 \rangle$
$\Delta_\mathbb{X} : A \to A \times A$	$\Delta : A \to A \otimes A$	$\Delta_\mathbb{X}^{(-1)}$

Table 3.1: Structural maps for the tensor in $Inv(\mathbb{X})$

The monoid coherence diagrams and Δ being total follow directly from the characteristics of the product in $\mathbb{X}$. It remains to show co-commutativity, co-associativity and the Frobenius condition.

Co-commutativity requires $\Delta c_\otimes = c_\otimes$. From the definitions, this means we need

$$\Delta_\mathbb{X} \langle \pi_1, \pi_0 \rangle = \Delta_\mathbb{X}.$$

Once again, this follows immediately from the definition of restriction product.

Co-associativity requires $\Delta(1 \otimes \Delta) = \Delta(\Delta \otimes 1)a_\otimes$. Expressing this in $\mathbb{X}$, we require

$$\Delta_\mathbb{X}(1 \times \Delta_\mathbb{X}) = \Delta_\mathbb{X}(\Delta_\mathbb{X} \times 1)(\langle \pi_0 \pi_0, \langle \pi_0 \pi_1, \pi_1 \rangle \rangle).$$

Again each is equal based on the properties of the restriction product.

The Frobenius requirement is two-fold:

$$\Delta^{(-1)} \Delta = (\Delta \otimes 1)a_\otimes (1 \otimes \Delta^{(-1)}) \tag{3.1}$$

$$\Delta^{(-1)} \Delta = (1 \otimes \Delta)a_\otimes{}^{(-1)}(\Delta^{(-1)} \otimes 1), \tag{3.2}$$

but in $\mathbb{X}$, this becomes:

$$\Delta_\mathbb{X}{}^{(-1)} \Delta_\mathbb{X} = (\Delta_\mathbb{X} \times 1)\langle \pi_0 \pi_0, \langle \pi_0 \pi_1, \pi_1 \rangle \rangle (1 \times \Delta_\mathbb{X}{}^{(-1)}) \tag{3.3}$$

$$\Delta_\mathbb{X}{}^{(-1)} \Delta_\mathbb{X} = (1 \times \Delta_\mathbb{X})\langle \langle \pi_0, \pi_1 \pi_0 \rangle, \pi_1 \pi_1 \rangle (\Delta_\mathbb{X}{}^{(-1)} \times 1). \tag{3.4}$$

We will detail the proof of equation equation (3.3) on the preceding page. Equation equation (3.4) on the previous page is proved similarly.

To show the equation, note first that $\Delta(1{\times}!)$ (and $\Delta(!\times 1)$) is the identity and secondly that maps to a product of objects may be split into a product map — e.g. if $f : A \to B \times B$, then $f = \langle f(1{\times}!), f(!\times 1)\rangle$.

Using this we see that the left hand side of equation equation (3.3) on the preceding page computes as follows:

$$\Delta_{\mathbb{X}}{}^{(-1)}\Delta_{\mathbb{X}} = \langle \Delta_{\mathbb{X}}{}^{(-1)}\Delta_{\mathbb{X}}(1{\times}!), \Delta_{\mathbb{X}}{}^{(-1)}\Delta_{\mathbb{X}}(!\times 1)\rangle$$
$$= \langle \Delta_{\mathbb{X}}{}^{(-1)}, \Delta_{\mathbb{X}}{}^{(-1)}\rangle$$

Similarly, removing the associativity maps, the right hand side of the same equation becomes:

$$(\Delta_{\mathbb{X}} \times 1)(1 \times \Delta_{\mathbb{X}}{}^{(-1)}) = \langle (\Delta_{\mathbb{X}} \times 1)(1 \times \Delta_{\mathbb{X}}{}^{(-1)})(1{\times}!), (\Delta_{\mathbb{X}} \times 1)(1 \times \Delta_{\mathbb{X}}{}^{(-1)})(!\times 1)\rangle$$
$$= \langle (\Delta_{\mathbb{X}} \times 1)(1 \times \Delta_{\mathbb{X}}{}^{(-1)})(1{\times}!), \Delta_{\mathbb{X}}{}^{(-1)}\rangle$$
$$= \langle (\Delta_{\mathbb{X}} \times 1)(1 \times \Delta_{\mathbb{X}}{}^{(-1)})(1 \times \Delta_{\mathbb{X}})(1{\times}!{\times}!), \Delta_{\mathbb{X}}{}^{(-1)}\rangle$$
$$= \langle (\Delta_{\mathbb{X}} \times 1)(1 \times \overline{\Delta_{\mathbb{X}}{}^{(-1)}})(1{\times}!{\times}!), \Delta_{\mathbb{X}}{}^{(-1)}\rangle$$
$$= \langle (\Delta_{\mathbb{X}} \times 1)\overline{1 \times \Delta_{\mathbb{X}}{}^{(-1)}}(1{\times}!{\times}!), \Delta_{\mathbb{X}}{}^{(-1)}\rangle$$
$$= \langle \overline{(\Delta_{\mathbb{X}} \times 1)(1 \times \Delta_{\mathbb{X}}{}^{(-1)})}(\Delta_{\mathbb{X}} \times 1)(1{\times}!{\times}!), \Delta_{\mathbb{X}}{}^{(-1)}\rangle$$
$$= \langle \overline{(\Delta_{\mathbb{X}} \times 1)(1 \times \Delta_{\mathbb{X}}{}^{(-1)})}(1{\times}!), \Delta_{\mathbb{X}}{}^{(-1)}\rangle$$
$$= \langle \overline{(\Delta_{\mathbb{X}} \times 1)(1 \times \Delta_{\mathbb{X}}{}^{(-1)})}(!\times 1)(1{\times}!), \Delta_{\mathbb{X}}{}^{(-1)}\rangle$$
$$= \langle \overline{\Delta_{\mathbb{X}}{}^{(-1)}}(1{\times}!), \Delta_{\mathbb{X}}{}^{(-1)}\rangle$$
$$= \langle \Delta_{\mathbb{X}}{}^{(-1)}\Delta_{\mathbb{X}}(1{\times}!), \Delta_{\mathbb{X}}{}^{(-1)}\rangle$$
$$= \langle \Delta_{\mathbb{X}}{}^{(-1)}, \Delta_{\mathbb{X}}{}^{(-1)}\rangle$$

and therefore we see that the first equation for the Frobenius condition is satisfied. Thus, $Inv(\mathbb{X})$ is a discrete inverse category. □

3.2 Completing a discrete inverse category

The purpose of this section is to prove that the category of discrete inverse categories is equivalent to the the category of discrete restriction categories. In order to prove this, we show how to construct a discrete restriction category, $\widetilde{\mathbb{X}}$, from a discrete inverse category, $\mathbb{X}$.

3.2.1 The restriction category $\widetilde{\mathbb{X}}$

Definition 3.2.1. When $\mathbb{X}$ is an inverse category, define $\widetilde{\mathbb{X}}$ as:

Objects: objects as in $\mathbb{X}$

Maps: equivalence classes of maps (the equivalence class is defined below in Definition 3.2.2 on the next page) with the following structure in $\mathbb{X}$:

$$\frac{A \xrightarrow{(f,C)} B \text{ in } \widetilde{\mathbb{X}}}{A \xrightarrow{f} B \otimes C \text{ in } \mathbb{X}}$$

Identity: by

$$\frac{A \xrightarrow{(u_{\otimes}^{r\,(-1)},1)} A}{A \xrightarrow{u_{\otimes}^{r\,(-1)}} A \otimes 1}$$

Composition: given by

$$\frac{A \xrightarrow{(f,B')} B \xrightarrow{(g,C')} C}{\dfrac{A \xrightarrow{f(g\otimes 1)a_{\otimes}} C \otimes (C' \otimes B')}{A \xrightarrow{(f(g\otimes 1)a_{\otimes},C'\otimes B')} C}}$$

When considering an $\widetilde{\mathbb{X}}$ map $(f,C) : A \to B$ in $\mathbb{X}$, we occasionally use the notation $f : A \to B_{|C}$ ($\equiv f : A \to B \otimes C$).

Equivalence classes of maps in $\mathbb{X}$

Definition 3.2.2. In a discrete inverse category $\mathbb{X}$ as defined above, the map f is equivalent to f' in $\mathbb{X}$ when $\overline{f} = \overline{f'}$ in $\mathbb{X}$ and the below diagram commutes for some map h:

Notation 3.2.3. When f is equivalent to g via the mediating map h, this is written as

$$f \stackrel{h}{\simeq} g.$$

Lemma 3.2.4. *Definition 3.2.2 gives a symmetric, reflexive equivalence class of maps in $\mathbb{X}$.*

Proof.

Reflexivity: Choose h as the identity map.

Symmetry: Suppose $f \stackrel{h}{\simeq} g$. Then, $\overline{f} = \overline{g}$ and $fk = g$ where $k = (\Delta \otimes 1)\, a_{\otimes}\, (1 \otimes h)\, a_{\otimes}^{(-1)}\, (\Delta^{(-1)} \otimes 1)$. Applying $k^{(-1)}$, which is $(\Delta \otimes 1)\, a_{\otimes}\, (1 \otimes h^{(-1)})\, a_{\otimes}^{(-1)}\, (\Delta^{(-1)} \otimes 1)$,

$$gk^{(-1)} = fkk^{(-1)} = f\overline{k} = \overline{fk}f = \overline{g}f = \overline{f}f = f.$$

Thus, $g \stackrel{h^{(-1)}}{\simeq} f$.

Transitivity: Suppose $f \stackrel{h}{\simeq} f'$ and $f' \stackrel{k}{\simeq} f''$. Then, consider the compositions of the mediating portions of the equivalences:

$$\ell = ((\Delta \otimes 1)a_{\otimes}(1 \otimes h)a_{\otimes}^{(-1)}(\Delta^{(-1)} \otimes 1))((\Delta \otimes 1)a_{\otimes}(1 \otimes k)a_{\otimes}^{(-1)}(\Delta^{(-1)} \otimes 1)).$$

47

By pasting the diagrams which give the above equivalences, we see that $f\ell = f''$. However, it is not in the form of a mediating map as presented.

The claim is that ℓ is the actual mediating map for f and f''. That is, that we have $f(\Delta \otimes 1)a_\otimes(1 \otimes \ell)a_\otimes{}^{(-1)}(\Delta^{(-1)} \otimes 1) = f''$. In the interest of some brevity, this is shown below with the associativity maps elided from the equations.

We need to show that $(\Delta \otimes 1)(1 \otimes \ell)(\Delta^{(-1)} \otimes 1) = \ell$.

$$(\Delta \otimes 1)(1 \otimes \ell)(\Delta^{(-1)} \otimes 1)$$

$$= (\Delta \otimes 1)(1 \otimes \Delta \otimes 1)(1 \otimes 1 \otimes h)(1 \otimes \Delta^{(-1)} \otimes 1))$$

$$(1 \otimes \Delta \otimes 1)(1 \otimes 1 \otimes k)(1 \otimes \Delta^{(-1)} \otimes 1)(\Delta^{(-1)} \otimes 1)$$

$$= (\Delta \otimes 1)(\Delta \otimes 1 \otimes 1)(1 \otimes 1 \otimes h)(1 \otimes \Delta^{(-1)} \otimes 1))$$

$$(1 \otimes \Delta \otimes 1)(1 \otimes 1 \otimes k)(\Delta^{(-1)} \otimes 1 \otimes 1)(\Delta^{(-1)} \otimes 1) \quad \text{co-associativity}$$

$$= (\Delta \otimes 1)(1 \otimes h)(\Delta \otimes 1 \otimes 1)(1 \otimes \Delta^{(-1)} \otimes 1))$$

$$(1 \otimes \Delta \otimes 1)(\Delta^{(-1)} \otimes 1 \otimes 1)(1 \otimes k)(\Delta^{(-1)} \otimes 1) \quad \text{Naturality}$$

$$= (\Delta \otimes 1)(1 \otimes h)(\Delta^{(-1)} \otimes 1)(\Delta \otimes 1))$$

$$(\Delta^{(-1)} \otimes 1)(\Delta \otimes 1)(1 \otimes k)(\Delta^{(-1)} \otimes 1) \quad \text{Frobenius}$$

$$= (\Delta \otimes 1)(1 \otimes h)(\Delta^{(-1)} \otimes 1)(\Delta \otimes 1)(1 \otimes k)(\Delta^{(-1)} \otimes 1) \quad \Delta \text{ Total}$$

$$= \ell$$

and therefore $f \overset{\ell}{\simeq} f''$. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\square$

Corollary 3.2.5. *If $\overline{f} = \overline{g}$ in $\mathbb{X}$, a discrete inverse category, and the diagram*

commutes for some h, then there is a h' such that $f \overset{h'}{\simeq} g$.

Proof. Consider

$$(\Delta \otimes 1)\, a_\otimes\, (1 \otimes (1 \otimes h))\, a_\otimes{}^{(-1)}(\Delta^{(-1)} \otimes 1)$$

$$= (\Delta \otimes 1)\, ((1 \otimes 1) \otimes h)\, a_\otimes a_\otimes{}^{(-1)}(\Delta^{(-1)} \otimes 1) \qquad \text{Naturality}$$

$$= (\Delta \otimes 1)\, ((1 \otimes 1) \otimes h)\, (\Delta^{(-1)} \otimes 1) \qquad \text{Isomorphism Inverse}$$

$$= (\Delta(1 \otimes 1)\Delta^{(-1)}) \otimes h \qquad \text{Naturality of } \otimes$$

$$= (1 \otimes h) \qquad \Delta\Delta^{(-1)} = 1$$

and therefore we can set $h' = 1 \otimes h$. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad \square$

Lemma 3.2.6. $\widetilde{\mathbb{X}}$ *as defined above is a category.*

Proof. The maps are well defined, as shown in lemma . The existence of the identity map is due to the tensor $\otimes$ being defined on $\mathbb{X}$, an inverse category, hence $u_\otimes^{r\,(-1)}$ is defined.

It remains to show the composition is associative and that $(u_\otimes^{r\,(-1)}, 1)$ acts as an identity in $\widetilde{\mathbb{X}}$.

Associativity: Consider

$$A \xrightarrow{(f,B')} B \xrightarrow{(g,C')} C \xrightarrow{(h,D')} D.$$

To show the associativity of this in $\widetilde{\mathbb{X}}$, we need to show in $\mathbb{X}$ that

$$\overline{(f(g \otimes 1)a_\otimes)(h \otimes 1)a_\otimes} = \overline{f(((g(h \otimes 1)a_\otimes) \otimes 1)a_\otimes)}$$

and that there exists a mediating map between the two of them.

To see that the restrictions are equal, first note that by the functorality of $\otimes$, for any two maps u and v, we have $uv \otimes 1 = (u \otimes 1)(v \otimes 1)$. Second, the naturality of $a_\otimes$ gives us that

$a_\otimes(h \otimes 1) = ((h \otimes 1) \otimes 1)a_\otimes$. Thus,

$$
\begin{aligned}
\overline{f(g \otimes 1)a_\otimes(h \otimes 1)a_\otimes} &= \overline{f(g \otimes 1)a_\otimes(h \otimes 1)\overline{a_\otimes}} && \text{Lemma 1.1.3}\\
&= \overline{f(g \otimes 1)a_\otimes(h \otimes 1)} && \overline{a_\otimes} = 1\\
&= \overline{f(g \otimes 1)((h \otimes 1) \otimes 1)a_\otimes} && a_\otimes \text{ natural}\\
&= \overline{f(g \otimes 1)((h \otimes 1) \otimes 1)} && a_\otimes \text{ iso, Lemma 1.1.3}\\
&= \overline{f(g \otimes 1)((h \otimes 1) \otimes 1)(a_\otimes \otimes 1)} && a_\otimes \otimes 1 \text{ iso, Lemma 1.1.3}\\
&= \overline{f((g(h \otimes 1)a) \otimes 1)} && \text{see above}\\
&= \overline{f((g(h \otimes 1)a) \otimes 1)a_\otimes} && a_\otimes \text{ iso}
\end{aligned}
$$

For the mediating map, see the diagram below, where calculation is in $\mathbb{X}$. The path starting at the top left at A and going right to $D_{|D' \otimes (C' \otimes B')}$ is grouping parentheses to the left, while starting in the same place but going down to $(D_{|D' \otimes C'})_{|B'}$ and then right to $D_{|(D' \otimes C') \otimes B'}$ is grouping parentheses to the right. The commutativity of the diagram is shown by the commutativity of the internal portions, which all follow from the standard coherence diagrams for the tensor and naturality of association.

From this, we can conclude

$$
(f(g \otimes 1)a_\otimes)(h \otimes 1)a_\otimes \overset{1 \otimes a_\otimes^{(-1)}}{\cong} f(((g(h \otimes 1)a_\otimes) \otimes 1)a_\otimes)
$$

which gives us that composition in $\widetilde{\mathbb{X}}$ is associative.

Identity: This requires:

$$(f,C)(u_\otimes^{r\,(-1)},1) = (f,C) = (u_\otimes^{r\,(-1)},1)(f,C)$$

for all maps $A \xrightarrow{(f,C)} B$ in $\widetilde{\mathbb{X}}$.

First, we see $\overline{f(u_\otimes^{r\,(-1)} \otimes 1)a_\otimes} = \overline{f}$ by Lemma 1.1.3 on page 3. Then, calculating in $\mathbb{X}$, we have a mediating map of $1 \otimes u_\otimes^l$ as shown below.

Next, $\overline{u_\otimes^{r\,(-1)}(f \otimes 1)a_\otimes} = \overline{f}$ by the naturality of $u_\otimes^{r\,(-1)}$ and Lemma 1.1.3 on page 3. The diagram below

shows our mediating map is $1 \otimes u_\otimes^r$. $\qquad\square$

Defining the restriction on $\widetilde{\mathbb{X}}$

Define the restriction in $\widetilde{\mathbb{X}}$ as follows:

$$\frac{A \xrightarrow{(f,C)} B}{\dfrac{A \xrightarrow{\overline{(f,C)}} A}{A \xrightarrow{\overline{f}u_\otimes^{r\,(-1)}} A \otimes 1 \text{ in } \mathbb{X}}}$$

Lemma 3.2.7. *The category $\widetilde{\mathbb{X}}$ with restriction defined as above is a restriction category.*

Proof. Given the above definition, the four restriction axioms must now be checked. (Diagrams are in $\mathbb{X}$).

[*R.1*] $(\overline{f}f = f)$ Calculating the restriction of the left hand side in $\mathbb{X}$, we have:

$$\overline{\overline{f}u_{\otimes}^{r\,(-1)}(f \otimes 1)a_{\otimes}} = \overline{\overline{f}u_{\otimes}^{r\,(-1)}(f \otimes 1)} \qquad\qquad a_{\otimes} \text{ iso, Lemma } 1.1.3$$

$$= \overline{\overline{f}fu_{\otimes}^{r\,(-1)}} \qquad\qquad u_{\otimes}^{r\,(-1)} \text{ natural}$$

$$= \overline{fu_{\otimes}^{r\,(-1)}} \qquad\qquad [\boldsymbol{R.1}] \text{ in } \mathbb{X}$$

$$= \overline{f} \qquad\qquad u_{\otimes}^{r\,(-1)} \text{ iso, Lemma } 1.1.3.$$

Then, the following diagram

shows $\overline{f}u_{\otimes}^{r\,(-1)}(f \otimes 1)a_{\otimes} \overset{1\otimes u_{\otimes}^{r}}{\simeq} f$ in $\mathbb{X}$ and therefore $\overline{f}f = f$ in $\widetilde{\mathbb{X}}$.

[*R.2*] $(\overline{g}\overline{f} = \overline{f}\overline{g})$ The restriction of the left hand side equals the restriction of the right hand side as seen below:

$$\overline{\overline{f}u_{\otimes}^{r\,(-1)}((\overline{g}u_{\otimes}^{r\,(-1)}) \otimes 1))a_{\otimes}} = \overline{\overline{f}(\overline{g}u_{\otimes}^{r\,(-1)})u_{\otimes}^{r\,(-1)}a_{\otimes}} \qquad u_{\otimes}^{r\,(-1)} \text{ natural}$$

$$= \overline{\overline{g}\overline{f}u_{\otimes}^{r\,(-1)}u_{\otimes}^{r\,(-1)}a_{\otimes}} \qquad [\boldsymbol{R.2}] \text{ in } \mathbb{X}$$

$$= \overline{\overline{g}u_{\otimes}^{r\,(-1)}((\overline{f}u_{\otimes}^{r\,(-1)}) \otimes 1)a_{\otimes}} \qquad u_{\otimes}^{r\,(-1)}\text{natural}.$$

52

The below diagram commutes by the naturality of $u_\otimes^r$ and the tensor coherence,

$$
\begin{array}{ccccccc}
A & \xrightarrow{\ \overline{g}u_\otimes^{r\,(-1)}\ } & A\otimes 1 & \xrightarrow{\ (\overline{f}u_\otimes^{r\,(-1)})\otimes 1\ } & (A\otimes 1)\otimes 1 & \xrightarrow{\ a_\otimes\ } & A\otimes(1\otimes 1)
\end{array}
$$

which allows us to conclude $\overline{f}\overline{g}=\overline{g}\overline{f}$ in $\widetilde{\mathbb{X}}$.

R.3 ($\overline{\overline{f}g}=\overline{f}\overline{g}$). As above, the first step is to show that the restrictions of each side are the same. Computing the restriction of the left hand side in $\mathbb{X}$:

$$
\begin{aligned}
\overline{\overline{(\overline{f}u_\otimes^{r\,(-1)})(g\otimes 1)a_\otimes}u_\otimes^{r\,(-1)}} &= \overline{\overline{(\overline{f}u_\otimes^{r\,(-1)})(g\otimes 1)a_\otimes}} && u_\otimes^{r\,(-1)}\ \text{iso, Lemma 1.1.3} \\
&= \overline{(\overline{f}u_\otimes^{r\,(-1)})(g\otimes 1)a_\otimes} && \text{Lemma 1.1.3} \\
&= \overline{\overline{f}gu_\otimes^{r\,(-1)}a_\otimes} && u_\otimes^{r\,(-1)}\ \text{natural} \\
&= \overline{\overline{f}g} && u_\otimes^{r\,(-1)}, a_\otimes\ \text{iso, Lemma 1.1.3} \\
&= \overline{f}\overline{g} && [\boldsymbol{R.3}]\ \text{in}\ \mathbb{X}.
\end{aligned}
$$

The restriction of the right hand side computes in $\mathbb{X}$ as:

$$
\begin{aligned}
\overline{(\overline{f}u_\otimes^{r\,(-1)})(\overline{g}u_\otimes^{r\,(-1)}\otimes 1)a_\otimes} &= \overline{(\overline{f}u_\otimes^{r\,(-1)})(\overline{g}u_\otimes^{r\,(-1)}\otimes 1)} && a_\otimes\ \text{iso, Lemma 1.1.3} \\
&= \overline{\overline{f}\overline{g}u_\otimes^{r\,(-1)}u_\otimes^{r\,(-1)}} && u_\otimes^{r\,(-1)}\ \text{natural} \\
&= \overline{\overline{f}\overline{g}} && u_\otimes^{r\,(-1)}u_\otimes^{r\,(-1)}\ \text{iso, Lemma 1.1.3} \\
&= \overline{f}\overline{g} && \text{Lemma 1.1.3.}
\end{aligned}
$$

Additionally, we see $\overline{\overline{f}g}$ in $\widetilde{\mathbb{X}}$ is expressed in $\mathbb{X}$ as:

$$
\begin{aligned}
\overline{(\overline{f}u_\otimes^{r\,(-1)})(g\otimes 1)a_\otimes}u_\otimes^{r\,(-1)} &= \overline{f}u_\otimes^{r\,(-1)}\overline{g\otimes 1} && [\boldsymbol{R.3}],[\boldsymbol{R.4}],a_\otimes\ \text{iso} \\
&= \overline{f}\overline{g}u_\otimes^{r\,(-1)} && \otimes\text{a restriction bi-functor},\ u_\otimes^{r\,(-1)}\ \text{natural.}
\end{aligned}
$$

53

The following diagram in $\mathbb{X}$ follows the above right hand side with the top curved arrow and the left hand side with the bottom curved arrow. Note that we are using that $\overline{(\overline{f}u_\otimes^{r\,(-1)})(g\otimes 1)a_\otimes} = \overline{f}\overline{g}$ as shown above.
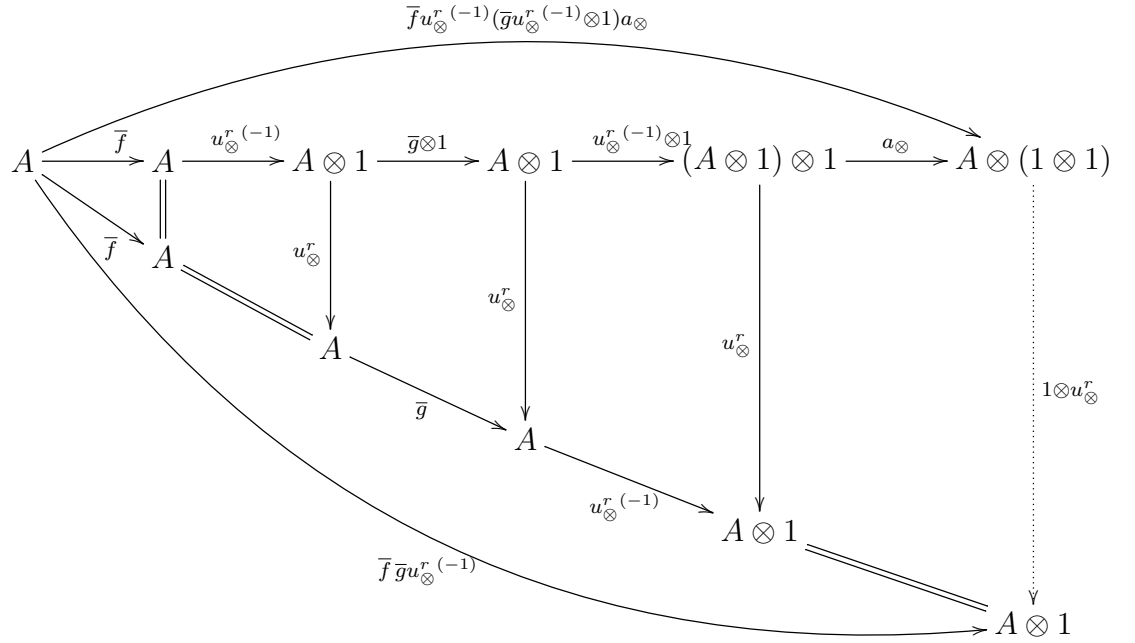
Hence, in $\mathbb{X}$, $\overline{(\overline{f}u_\otimes^{r\,(-1)})(g\otimes 1)a_\otimes}\,u_\otimes^{r\,(-1)} \overset{1\otimes u_\otimes^r}{\simeq} (\overline{f}u_\otimes^{r\,(-1)})(\overline{g}u_\otimes^{r\,(-1)}\otimes 1)a_\otimes$ and therefore $\overline{\overline{f}g} = \overline{f}\overline{g}$ in $\widetilde{\mathbb{X}}$.

R.4 $f\overline{g} = \overline{f}\overline{g}f$ The restriction of the left hand side is:

$$
\begin{aligned}
\overline{f(\overline{g}u_\otimes^{r\,(-1)}\otimes 1)a_\otimes} &= \overline{f(\overline{g}u_\otimes^{r\,(-1)}\otimes 1)} && a_\otimes \text{ iso, Lemma 1.1.3} \\
&= \overline{f\overline{g}u_\otimes^{r\,(-1)}}\otimes \overline{f} && \otimes \text{ restriction functor} \\
&= \overline{f\overline{g}}\otimes \overline{f} && u_\otimes^{r\,(-1)} \text{ iso, Lemma 1.1.3} \\
&= \overline{f(\overline{g}\otimes 1)}
\end{aligned}
$$

and the restriction of the right hand side is:

$$\overline{\overline{f(g \otimes 1)}u_\otimes^{r\,(-1)}}(f \otimes 1)a_\otimes = \overline{\overline{f(g \otimes 1)}u_\otimes^{r\,(-1)}}(f \otimes 1) \qquad a_\otimes \text{ iso, Lemma } 1.1.3$$

$$= \overline{\overline{f(g \otimes 1)}fu_\otimes^{r\,(-1)}} \qquad\qquad u_\otimes^{r\,(-1)} \text{ natural}$$

$$= \overline{\overline{f}(g \otimes 1)u_\otimes^{r\,(-1)}} \qquad\qquad [\boldsymbol{R.4}] \text{ for } \mathbb{X}$$

$$= \overline{f(\overline{g} \otimes 1)u_\otimes^{r\,(-1)}} \qquad\qquad \otimes \text{ is a restriction functor}$$

$$= \overline{f(\overline{g} \otimes 1)} \qquad\qquad u_\otimes^{r\,(-1)} \text{ iso, Lemma } 1.1.3$$

Computing the right hand side in $\mathbb{X}$,

$$\overline{f(g \otimes 1)a_\otimes}u_\otimes^{r\,(-1)}(f \otimes 1)a_\otimes = \overline{f(g \otimes 1)}fu_\otimes^{r\,(-1)}a_\otimes \qquad a_\otimes \text{ iso, } u_\otimes^{r\,(-1)} \text{ natural.}$$

$$= f(\overline{g} \otimes 1)u_\otimes^{r\,(-1)}a_\otimes \qquad [\boldsymbol{R.3}], \otimes \text{ a restriction functor.}$$

$$
\begin{array}{ccccccc}
A & \xrightarrow{\;f\;} & B \otimes C & \xrightarrow{\;\overline{g}u_\otimes^{r\,(-1)} \otimes 1\;} & (B \otimes 1) \otimes C & \xrightarrow{\;a_\otimes\;} & B \otimes (1 \otimes C) \\
& \searrow{\scriptstyle f} & & & & & \bigg\downarrow{\scriptstyle 1 \otimes c_\otimes} \\
& & B \otimes C & \xrightarrow{\;\overline{g} \otimes 1\;} B \otimes C \xrightarrow{\;u_\otimes^{r\,(-1)}\;} & (B \otimes C) \otimes 1 & \xrightarrow{\;a_\otimes\;} & B \otimes (C \otimes 1)
\end{array}
$$

and hence, $\widetilde{\mathbb{X}}$ is a restriction category. $\qquad\qquad\square$

3.2.2 The category $\widetilde{\mathbb{X}}$ is a discrete restriction category

Lemma 3.2.8. *The unit of the inverse product in $\mathbb{X}$ is the terminal object in $\widetilde{\mathbb{X}}$.*

Proof. The unique map to the terminal object for any object A in $\widetilde{\mathbb{X}}$ is the equivalence class of maps represented by $(u_\otimes^{l\,(-1)}, A)$. For this to be a terminal object, the diagram

$$
\begin{array}{ccc}
X & \xrightarrow{\;\overline{(f,C)}\;} X \xrightarrow{\;!_X\;} & \top \\
\bigg\downarrow{\scriptstyle (f,C)} & \nearrow{\scriptstyle !_Y} & \\
Y & &
\end{array}
$$

must commute for all choices of f. Translating this to $\mathbb{X}$, this is the same as requiring

$$
\begin{array}{ccccc}
X & \xrightarrow{\;\overline{f}\;} & X & \xrightarrow{\;u_\otimes^{r\,(-1)}\;} X \otimes 1 & \xrightarrow{\;u_\otimes^{l\,(-1)}\;} 1 \otimes X \otimes 1 \\
\bigg\downarrow{\scriptstyle f} & & & \nwarrow{\scriptstyle 1 \otimes (u_\otimes^r f)} & \\
Y \otimes C & & \xrightarrow{\;u_\otimes^{l\,(-1)}\;} & 1 \otimes Y \otimes C &
\end{array}
$$

55

commute, which is true by [**R.1**] and from the coherence diagrams for the inverse product tensor. □

Next, we show that the category $\widetilde{\mathbb{X}}$ has restriction products, given by the the action of $\widetilde{(\_)}$ on the $\otimes$ tensor in $\mathbb{X}$.

First, define total maps π_0, π_1 in $\widetilde{\mathbb{X}}$ by:

$$\pi_0: \quad A \otimes B \xrightarrow{(1,B)} A \tag{3.5}$$

$$\pi_1: \quad A \otimes B \xrightarrow{(c_\otimes, A)} B \tag{3.6}$$

Given the maps $Z \xrightarrow{(f,C)} A$ and $Z \xrightarrow{(g,C')} B$, define $\langle (f,C),(g,C')\rangle$ as

$$Z \xrightarrow{(\Delta(f\otimes g)(1\otimes c_\otimes \otimes 1), C\otimes C')} A \otimes B \tag{3.7}$$

where associativity is assumed as needed. Note that with the associativity maps, this is actually:

$$Z \xrightarrow{(\Delta(f\otimes g)a_\otimes(1\otimes a_\otimes{}^{(-1)})(1\otimes(c_\otimes \otimes 1))(1\otimes a_\otimes)a_\otimes{}^{(-1)}, C\otimes C')} A \otimes B \tag{3.8}$$

Lemma 3.2.9. *On $\widetilde{\mathbb{X}}$, $\otimes$ is a restriction product with projections π_0, π_1 with the product of maps f, g being $\langle f, g\rangle$.*

Proof. From the definition above, as 1 and $c_\otimes$ are isomorphisms, the maps π_0, π_1 are total.

In order to show that $\overline{\langle f, g\rangle} = \overline{f}\,\overline{g}$, first reduce the left hand side:

$$
\begin{aligned}
\overline{\langle f, g\rangle} &= \overline{\Delta(f\otimes g)(1\otimes c_\otimes \otimes 1)u_\otimes^{r\,(-1)}} && \text{in } \mathbb{X}, \text{ definition of restriction} \\
&= \overline{\Delta(f\otimes g)u_\otimes^{r\,(-1)}} && c_\otimes \text{ is iso} \\
&= \overline{\Delta\overline{(f\otimes g)}u_\otimes^{r\,(-1)}} && \text{from Lemma 1.1.3} \\
&= \overline{\Delta(\overline{f}\otimes \overline{g})u_\otimes^{r\,(-1)}} && \otimes \text{ is a restriction functor} \\
&= \overline{\overline{f}\,\overline{g}\,\Delta(1\otimes 1)u_\otimes^{r\,(-1)}} && \text{Lemma 3.1.8((ii)) twice} \\
&= \overline{\overline{f}\,\overline{g}u_\otimes^{r\,(-1)}} && \text{Lemma 1.1.3} \\
&= \overline{f}\,\overline{g}u_\otimes^{r\,(-1)} && \text{Lemma 1.1.3.}
\end{aligned}
$$

Then, the right hand side reduces as:

$$\overline{f}\overline{g} = \overline{f}u_{\otimes}^{r\,(-1)}(\overline{g}u_{\otimes}^{r\,(-1)} \otimes 1)a_{\otimes} \qquad\qquad \text{in } \mathbb{X} \text{ by definitions}$$

$$= \overline{f}\overline{g}u_{\otimes}^{r\,(-1)}u_{\otimes}^{r\,(-1)}a_{\otimes} \qquad\qquad u_{\otimes}^{r\,(-1)} \text{ natural.}$$

The restriction of the left hand side and the right hand side, in $\mathbb{X}$, is $\overline{\overline{f}\overline{g}}$. This is done by applying Lemma 1.1.3 on page 3 once on the left and thrice on the right.

Thus, this shows $\overline{\langle f, g\rangle} = \overline{f}\overline{g}$ in $\widetilde{\mathbb{X}}$ where the mediating map in $\mathbb{X}$ is $1 \otimes u_{\otimes}^{r}$.

Next, to show $\langle f, g\rangle\pi_0 \leq f$ (and $\langle f, g\rangle\pi_1 \leq g$), it is required to show $\overline{\langle f, g\rangle\pi_0}f = \langle f, g\rangle\pi_0$. Calculating the left side, we see:

$$\overline{\langle f, g\rangle\pi_0}f = \overline{\langle f, g\rangle\overline{\pi_0}}f \qquad\qquad \text{Lemma 1.1.3}$$

$$= \overline{\langle f, g\rangle}f \qquad\qquad \pi_0 \text{ is total}$$

$$= \overline{f}\,\overline{g}\,f \qquad\qquad \text{by above}$$

$$= \overline{g}\overline{f}f \qquad\qquad [\boldsymbol{R.2}]$$

$$= \overline{g}f \qquad\qquad [\boldsymbol{R.1}].$$

Now, turning to the right hand side:

$$\langle f, g\rangle\pi_0 = \Delta(f \otimes g)(1 \otimes c_{\otimes} \otimes 1)1 \qquad\qquad \text{in } \mathbb{X}, \text{ by definition.}$$

To show these are equal in $\widetilde{\mathbb{X}}$, we need to first show the restrictions are the same in $\mathbb{X}$ and then show there is a mediating map between the images in $\mathbb{X}$. The restriction of $\overline{g}f$ is $\overline{f}\overline{g}$ immediately by $[\boldsymbol{R.3}]$ and $[\boldsymbol{R.2}]$. For the right hand side, calculate in $\mathbb{X}$:

$$\overline{\Delta(f \otimes g)(1 \otimes c_{\otimes} \otimes 1)} = \overline{\Delta(f \otimes g)} \qquad\qquad \text{Lemma 1.1.3}$$

$$= \Delta(f \otimes g)(f^{(-1)} \otimes g^{(-1)})\Delta^{(-1)} \qquad \mathbb{X} \text{ is an inverse category}$$

$$= \Delta(\overline{f} \otimes \overline{g})\Delta^{(-1)}$$

$$= \overline{f}\overline{g}\Delta\Delta^{(-1)} \qquad\qquad \text{Lemma 3.1.8((ii)) twice}$$

$$= \overline{f}\overline{g}.$$

The diagram below, shows the required mediating map.

$$
\begin{array}{c}
\xymatrix{
& & & A \otimes C \ar[drr]^{\Delta \otimes 1} & & \\
& & Z \ar[urr]^{f} & & & A \otimes A \otimes C \ar@{.>}[d]^{1 \otimes \Delta} \\
Z \ar[ur]^{\overline{g}} \ar[dr]_{\Delta} & & & & & A \otimes A \otimes C \otimes A \otimes C \\
}
\end{array}
$$

The diagram (mediating map) shows:

- $A \otimes C \xrightarrow{\Delta \otimes 1} A \otimes A \otimes C$
- $Z \xrightarrow{f} A \otimes C$
- $Z \xrightarrow{\overline{g}} Z$
- $Z \xrightarrow{\Delta} Z \otimes Z$
- $A \otimes A \otimes C \xrightarrow{1 \otimes \Delta} A \otimes A \otimes C \otimes A \otimes C$
- $A \otimes A \otimes C \otimes A \otimes C \xrightarrow{1 \otimes 1 \otimes f^{(-1)}} A \otimes A \otimes C \otimes Z$
- $A \otimes A \otimes C \otimes Z \xrightarrow{1 \otimes 1 \otimes 1 \otimes g} A \otimes A \otimes C \otimes B \otimes C'$
- $A \otimes A \otimes C \otimes B \otimes C' \xrightarrow{1 \otimes 1 \otimes c_{\otimes} \otimes 1} A \otimes A \otimes B \otimes C \otimes C'$
- $Z \otimes Z \xrightarrow{f \otimes g} A \otimes C \otimes B \otimes C'$
- $A \otimes C \otimes B \otimes C' \xrightarrow{1 \otimes c_{\otimes} \otimes 1} A \otimes B \otimes C \otimes C'$
- $A \otimes A \otimes B \otimes C \otimes C' \xrightarrow{\Delta^{(-1)} \otimes 1 \otimes 1 \otimes 1} A \otimes B \otimes C \otimes C'$

$\square$
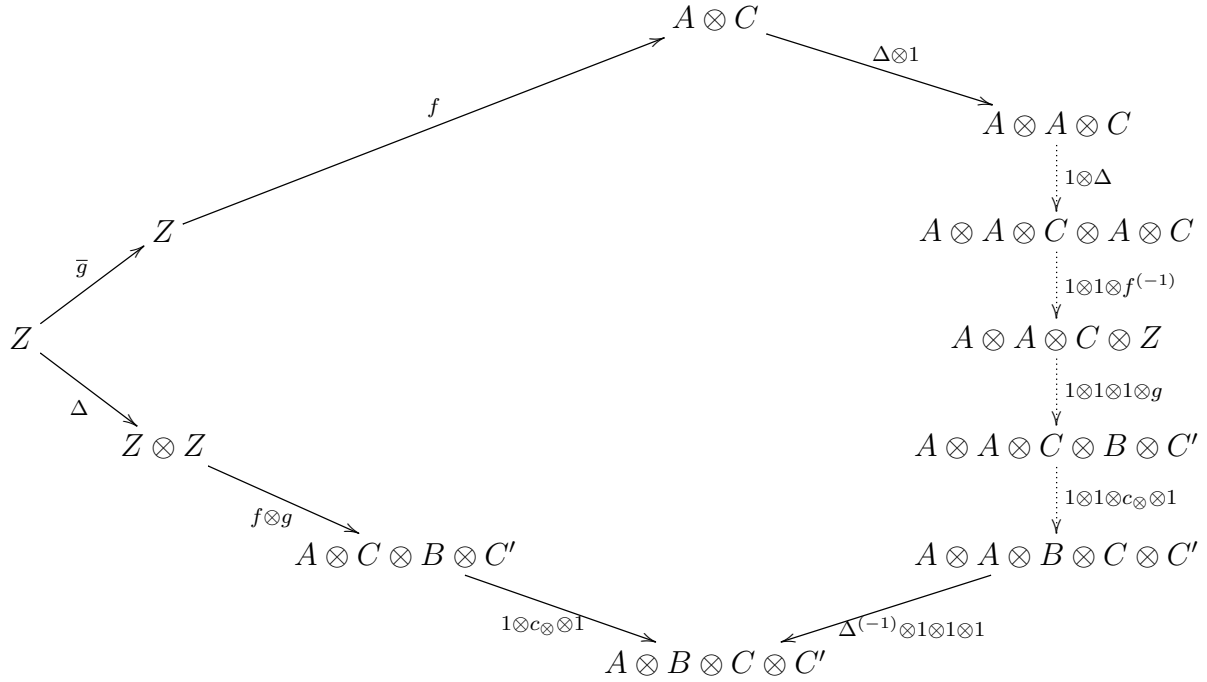
At this point, we have shown that $\widetilde{\mathbb{X}}$ is a restriction category with restriction products. This leads us to the following theorem:

Theorem 3.2.10. *For any inverse category $\mathbb{X}$, the category $\widetilde{\mathbb{X}}$ is a discrete restriction category.*

Proof. The fact that $\widetilde{\mathbb{X}}$ is a Cartesian restriction category is immediate from lemmas 3.2.6 on page 49, 3.2.7 on page 52, 3.2.8 on page 55 and 3.2.9 on page 56.

To show that it is discrete, we need only show that the map $(\Delta u_{\otimes}^{r}{}^{(-1)}, 1)$ is in the same equivalence class as $\widetilde{\mathbb{X}}$'s $\Delta (= \langle 1, 1 \rangle = \langle (u_{\otimes}^{r}{}^{(-1)}, 1), (u_{\otimes}^{r}{}^{(-1)}, 1) \rangle$. As both Δ and $u_{\otimes}^{r}{}^{(-1)}$ are total, the restriction of each side is the same, namely 1. The diagram below uses Corollary

and shows that the two maps are in the same equivalence class.

$$
\begin{array}{ccc}
 & & A \otimes A \otimes 1 \\
 & \overset{\Delta u_\otimes^{r\,(-1)}}{\nearrow} & \Big\downarrow {\scriptstyle u_\otimes^{r\,(-1)}} \\
A & \xrightarrow[\Delta(u_\otimes^{r\,(-1)} \otimes u_\otimes^{r\,(-1)})(1 \otimes c_\otimes \otimes 1)]{} & A \otimes A \otimes 1 \otimes 1
\end{array}
$$

$\square$

3.2.3 Equivalence of categories

This section will show that the category of discrete inverse categories (maps being restriction functors that preserve the inverse tensor) is equivalent to the category of discrete restriction categories (maps being the restriction functors which preserve the product). In the following, $\mathbb{X}$ will always be a discrete inverse category, $\mathbb{D}$ and $\mathbb{C}$ will be discrete restriction categories.

We approach the equivalence proof by exhibiting the universal property for discrete inverse categories for the functor **INV** from discrete restriction categories to discrete inverse categories. The functor **INV** maps a discrete restriction category to its inverse subcategory and maps functors between discrete restriction categories to a functor having the same action on the partial inverses. That is, given $G : \mathbb{C} \to \mathbb{D}$, then:

$\mathbf{INV}(G) : \mathbf{INV}(\mathbb{C}) \to \mathbf{INV}(\mathbb{D})$

$\mathbf{INV}(G)(A) = GA$ $\qquad\qquad\qquad$ (all objects of $\mathbb{D}$ are in $Inv(\mathbb{D})$)

$\mathbf{INV}(G)(f) = G(f)$ $\qquad\qquad$ (restriction functors preserve partial inverse)

We continue by showing the η and ε of the universal property are isomorphisms. First, let $\eta : \mathbb{X} \to \mathbf{INV}(\widetilde{\mathbb{X}})$ be an identity on objects functor. For maps f in $\mathbb{X}$, $\eta(f) = (fu_\otimes^{r\,(-1)}, 1)$.

Next, consider a functor $F : \mathbb{X} \to \mathbf{INV}(\mathbb{D})$ defined as follows:

Objects: $F^{\#} : A \mapsto F(A)$

Arrows: $F^{\#} : (f, C) \mapsto F(f)\pi_0$

This allows us to write the diagram:

$$
\begin{array}{ccc}
\mathbb{X} & \xrightarrow{\quad \eta \quad} & \mathbf{INV}(\widetilde{\mathbb{X}}) \\
 & F \searrow & \downarrow \mathbf{INV}(F^{\#}) \\
 & & \mathbf{INV}(\mathbb{D})
\end{array}
\tag{3.9}
$$

In order to show this is a universal diagram, we proceed with a series of lemmas building to the result.

Lemma 3.2.11. *For any discrete inverse category $\mathbb{X}$, all invertible maps $(g, C) : A \to B$ in $\widetilde{\mathbb{X}}$ are in the equivalence class of $(f u_\otimes^{r\,(-1)}, 1)$ for some $f : A \to B$.*

Proof. As (g, C) is invertible in $\widetilde{\mathbb{X}}$, the map $(g, C)^{(-1)} : B \to A$ exists. $(g, C)^{(-1)}$ must be in the equivalence class of some map $k : B \to A \otimes D$, and also note that $\overline{(g, C)}$ is by construction the equivalence class of the map $\overline{g} u_\otimes^{r\,(-1)} : A \to A \otimes 1$ in $\mathbb{X}$. This means, diagramming in $\mathbb{X}$, there is an n such that

$$
\begin{array}{ccc}
B & \xrightarrow{\quad k \quad} A \otimes D \xrightarrow{\quad f \otimes 1 \quad} & B \otimes C \otimes D \\
 & & \downarrow {\Delta \otimes 1} \\
 & & B \otimes B \otimes C \otimes D \\
 & \overline{k} u_\otimes^{r\,(-1)} & \vdots {\scriptstyle 1 \otimes n} \\
 & & B \otimes B \otimes 1 \\
 & & \downarrow {\Delta^{(-1)} \otimes 1} \\
 & \searrow & B \otimes 1
\end{array}
$$

commutes.

Starting with $g : A \to B \otimes C$, construct the map f in $\mathbb{X}$ with the following diagram:

$$
\begin{array}{ccc}
A & \xrightarrow{\quad g \quad} & B \otimes C \\
& & \downarrow{\scriptstyle \Delta \otimes 1} \\
& & B \otimes B \otimes C \\
& & \downarrow{\scriptstyle 1 \otimes \Delta \otimes 1} \\
& & B \otimes B \otimes B \otimes C \\
& & \downarrow{\scriptstyle 1 \otimes 1 \otimes k \otimes 1} \\
& & B \otimes B \otimes A \otimes D \otimes C \\
& & \downarrow{\scriptstyle 1 \otimes 1 \otimes g \otimes 1 \otimes 1} \\
f & & B \otimes B \otimes B \otimes C \otimes D \otimes C \\
& & \downarrow{\scriptstyle 1 \otimes \Delta^{(-1)} \otimes 1 \otimes c_\otimes} \\
& & B \otimes B \otimes C \otimes C \otimes D \\
& & \downarrow{\scriptstyle 1 \otimes 1 \otimes \Delta^{(-1)} \otimes 1} \\
& & B \otimes B \otimes C \otimes D \\
& & \downarrow{\scriptstyle 1 \otimes n} \\
& & B \otimes B \otimes 1 \\
& & \downarrow{\scriptstyle (\Delta^{(-1)} \otimes 1) u^l_\otimes} \\
& & B
\end{array}
$$

By its construction, $f : A \to B$ in $\mathbb{X}$ and $(f u_\otimes^{r\,(-1)}, 1)$ is in the same equivalence class as (g, C).

$\square$

Lemma 3.2.12. *Diagram (equation (3.9)) above is a commutative diagram.*

Proof. Chasing maps around the diagram, we have:

$$
\begin{array}{ccc}
f & \xmapsto{\quad \eta \quad} & (f u_\otimes^{r\,(-1)}, 1) \\
& \searrow{\scriptstyle F} & \big\uparrow{\scriptstyle \mathbf{INV}(F^\#)} \\
& F(f) =\!=\!= & F(f u_\otimes^{r\,(-1)}) \pi_0
\end{array}
$$

As η is identity on the objects, diagram equation (3.9) on the previous page commutes. $\square$

Lemma 3.2.13. *The functor* **INV** *from the category of discrete restriction categories to the category of discrete inverse categories is full and faithful.*

Proof. To show fullness, we must show **INV** is surjective on hom-sets. Given a functor between two categories in the image of **INV**, i.e., $G : \mathbf{INV}(\mathbb{C}) \to \mathbf{INV}(\mathbb{D})$, construct a functor $H : \mathbb{C} \to \mathbb{D}$ as follows:

Action on objects: $H(A) = G(A)$,

Objects on maps: $H(f) = G(\langle f, 1 \rangle)\pi_0$.

H is well defined as we know $\langle f, 1 \rangle$ is an invertible map and therefore in the domain of G. To see H is a functor:

$$H(1) = G(\langle 1, 1 \rangle)\pi_0 = \Delta_{\mathbb{D}}\pi_0 = 1$$

$$H(fg) = G(\langle fg, 1 \rangle)\pi_0 = G(\langle f, 1 \rangle)\pi_0 G(\langle g, 1 \rangle)\pi_0 = H(f)H(g)$$

But on any invertible map, $H(f) = G(\langle f, 1 \rangle)\pi_0 = \langle G(f), 1 \rangle\pi_0 = G(f)$ and therefore $\mathbf{INV}(()H) = G$, so **INV** is full.

Next, assume we have $F, G : \mathbb{C} \to \mathbb{D}$ with $\mathbf{INV}(F) = \mathbf{INV}(G)$. Considering $F(f)$ and $F(g)$, we know $F(\langle f, 1 \rangle) = G(\langle f, 1 \rangle)$ as $\langle f, 1 \rangle$ is invertible. Thus, as the functors preserve the product structure, we have

$$F(f) = F(\langle f, 1 \rangle)F(\pi_0) = G(\langle f, 1 \rangle)G(\pi_0) = G(f).$$

Thus, **INV** is faithful. $\square$

Corollary 3.2.14. *The functor $F^{\#}$ in diagram equation (3.9) on the preceding page is unique.*

Proof. This follows immediately from lemma 3.2.13 on the previous page, **INV** is faithful.

$\square$

Corollary 3.2.15. *The category $\widetilde{\mathbb{X}}$ and functor $\eta : \mathbb{X} \to \mathbf{INV}(\widetilde{\mathbb{X}})$ is a universal pair for the functor* **INV***.*

Proof. Immediate from Corollary 3.2.14 and Lemma 3.2.12 on the previous page. $\square$

Lemma 3.2.16. *The functor $\eta : \mathbb{X} \to \mathbf{INV}(\widetilde{\mathbb{X}})$ is an isomorphism.*

Proof. As η is an identity on objects functor, we need only show that it is full and faithful. Referring to Lemma 3.2.11 on page 60 above, we immediately see that η is full. For faithful, if we assume $(fu_{\otimes}^{r\,(-1)}, 1)$ is equal in $\widetilde{\mathbb{X}}$ to $(gu_{\otimes}^{r\,(-1)}, 1)$. This means in $\mathbb{X}$, that $\overline{f} = \overline{g}$ and there is a h such that

This simplifies out to $g = f\Delta(1 \otimes h)\Delta^{(-1)}$. But by Lemma 3.1.8 on page 38, part (iv) on page 38, $\Delta(1 \otimes h)\Delta^{(-1)} = \overline{\Delta(1 \otimes h)\Delta^{(-1)}}$. Setting $\Delta(1 \otimes h)\Delta^{(-1)}$ as k, we have $g = f\overline{k}$. But this gives us:

$$g = f\overline{k} = \overline{fk}f = \overline{f}\overline{k}f = \overline{g}f = \overline{f}f = f.$$

This shows η is faithful and hence an isomorphism between $\mathbb{X}$ and $\mathbf{INV}(\widetilde{\mathbb{X}})$. $\qquad\square$

Theorem 3.2.17. *The category of discrete inverse categories (objects are discrete inverse categories, maps are inverse tensor preserving functors) is equivalent to the category of discrete restriction categories (objects are discrete restriction categories, maps are the Cartesian restriction functors).*

Proof. From the above lemmas, we have shown that we have an adjoint:

$$(\eta, \varepsilon) : \mathbf{T} \vdash \mathbf{INV} : D_{ic} \to D_{rc} \tag{3.10}$$

By lemma 3.2.16 we know η is an isomorphism. But this means the functor $\mathbf{T}$ is full and faithful, as shown in, e.g., Proposition 2.2.6 of [18]. From lemma 3.2.13 we know that $\mathbf{INV}$

is full and faithful. But again by the previous reference, this means ε is an isomorphism. Thus, by Corollary 3.2.15 and Proposition 2.2.7 of [18] we have the equivalence of the two categories. $\square$

3.2.4 Examples of the $\widetilde{(\_)}$ construction

Example 3.2.18 (Completing a finite discrete inverse category)**.**

Continuing from example 3.1.7 on page 37, recall the discrete category of 4 elements with two different tensors. Completing these gives two different lattices. They are either the straight line lattice, or the diamond semilattice. Below are the details of these constructions.

Recall $\mathbb{D}$ has four elements a, b, c and d, and there are two possible inverse product tensors:

$\otimes$	a	b	c	d
a	a	a	a	a
b	a	b	**b**	b
c	a	**b**	c	c
d	a	b	c	d

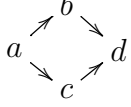| $\otimes$	a	b	c	d
a	a	a	a	a
b	a	b	**a**	b
c	a	**a**	c	c
d	a	b	c	d

Define Δ as the identity map. Then, for the first tensor, $\widetilde{\mathbb{D}}$ has the following maps

$$a \xrightarrow{(id,a) \ (\equiv(id,b)\equiv(id,c)\equiv(id,d))} a, \qquad a \xrightarrow{(id,a)} b, \qquad a \xrightarrow{(id,a)} c, \qquad a \xrightarrow{(id,a)} d$$

$$b \xrightarrow{(id,b) \ (\equiv(id,c)\equiv(id,d))} b, \qquad b \xrightarrow{(id,b)} c, \qquad b \xrightarrow{(id,b)} d$$

$$c \xrightarrow{(id,c) \ (\equiv(id,d))} c, \qquad c \xrightarrow{(id,c)} d$$

$$d \xrightarrow{(id,d)} d$$

resulting in the straight-line $(a \to b \to c \to d)$ lattice. The tensor in $\mathbb{D}$ becomes the meet and hence is a categorical product in $\widetilde{\mathbb{D}}$. Note that the only partial inverses in $\widetilde{\mathbb{D}}$ are the identity functions and that for all maps f, $\langle f, 1 \rangle = id$.

With the second tensor table, we have:

$$a \xrightarrow{(id,a)\ (\equiv(id,b)\equiv(id,c)\equiv(id,d))} a, \qquad a \xrightarrow{(id,a)} b, \qquad a \xrightarrow{(id,a)} c, \qquad a \xrightarrow{(id,a)} d$$

$$b \xrightarrow{(id,b)\ (\equiv(id,d))} b, \qquad b \xrightarrow{(id,b)} d$$

$$c \xrightarrow{(id,c)\ (\equiv(id,d))} c, \qquad c \xrightarrow{(id,c)} d$$

$$d \xrightarrow{(id,d)} d$$

resulting in the "diamond" lattice,
$$
\begin{array}{ccc}
 & b & \\
\nearrow & & \searrow \\
a & & d \\
\searrow & & \nearrow \\
 & c &
\end{array}
$$
. Once again, the tensor in $\mathbb{D}$ is the meet.

Example 3.2.19. Lattice completion. Suppose we have a set together with an idempotent, commutative, associative operation $\wedge$ on the set, giving us a lattice, $\mathbb{L}$. Further suppose the set is partially ordered via $\leq$ with the order being compatible with $\wedge$.

Then, we may create a pullback square for any $x' \leq x$, $y' \leq x$ with
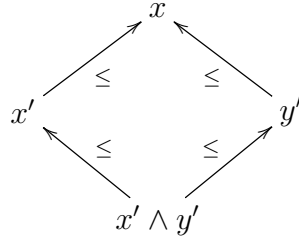
Considering $\mathbb{L}$ as a category, we see that all maps are monic and therefore, we may create a partial map category $\mathrm{Par}(\mathbb{L}, \mathcal{M})$ where the stable system of monics are all the maps.

Then $\widetilde{\mathrm{Par}(\mathbb{L}, \mathcal{M})}$ becomes the completion of the lattice over $\wedge$.

3.2.5 Quantum computation

Quantum computation proceeds via the application of reversible transformations — Unitary transformations.

The semantics of quantum computation can be defined as a †-compact closed category as introduced in [2, 3] and completely positive maps as discussed in [23].

Definition 3.2.20 (Dagger Category)**.** A *Dagger Category*[23] is a category $\mathbb{C}$ together with an operation † that is an involutive, identity on objects, contra-variant endofunctor on $\mathbb{C}$.

Recalling first that a *symmetric monoidal category* is a category $\mathbb{B}$ with a bi-functor $\otimes$, an object I and natural isomorphisms:

$$a_{A,B,C} : (A \otimes B) \otimes C \to A \otimes (B \otimes C)$$

$$c_{A,B} : A \otimes B \to B \otimes A$$

$$ul_A : A \to I \otimes A$$

with standard coherence conditions, as in [14]. Note that we also have a map $ur_A : A \to A \otimes I$ given by $ur_A = ul_A c_{I,A}$ Furthermore, a *compact closed category* $\mathbb{C}$ is a symmetric monoidal category where each object A has a dual A^* together with the maps:

$$\eta_A : I \to A^* \otimes A$$

$$\epsilon_A : A \otimes A^* \to I$$

such that

From the above, we can define a *Dagger symmetric monoidal category* and a *Dagger compact closed category*. The latter is referred to as a *strongly compact closed category* in [2], where they were initially introduced. In each case, the † functor is added in a way that retains coherence with the bi-functor $\otimes$ and with the dualizing operator. The coherence implies that the $i^\dagger = i^{-1}$ for the SMC isomorphisms, that $(f \otimes g)^\dagger = f^\dagger \otimes g^\dagger$ for all maps

f, g in the symmetric monoidal category and that

$$
\begin{array}{ccc}
I & \xrightarrow{\epsilon_A{}^\dagger} & A \otimes A^* \\
& {\eta_A} \searrow & \downarrow{c} \\
& & A^* \otimes A
\end{array}
$$

commutes for all objects A in the compact closed category.

Example 3.2.21 (REL)**.** REL is a dagger compact closed category with the dual of an object A is A, $\otimes$ is the cartesian product and for $R : A \to B$, we have $R^* = R^\dagger = \{(y, x) | (x, y) \in R\}$.

Example 3.2.22 (FDHILB)**.** The category of finite dimensional Hilbert spaces, FDHILB is a dagger compact closed category with the dual of an object H is the normal Hilbert space dual H^*, the space of continuous linear functions from H to the base field. $\otimes$ is the normal Hilbert space tensor and and for $f : A \to B$, we have $f^\dagger$ is the unique map such that $\langle fx|y \rangle = \langle y|f^\dagger x \rangle$ for all $x \in A$, $y \in B$.

Additionally, if one has a dagger compact closed category with biproducts where the biproducts and dagger interact such that $p_i{}^\dagger = q_i$, this is called a *biproduct dagger compact closed category.*

In [23], the author continues from this point: Starting with a biproduct dagger compact closed category $\mathbb{C}$, he creates a new category, CPM($\mathbb{C}$) which has the same objects as $\mathbb{C}$, but morphisms $f : A \to B$ in CPM($\mathbb{C}$) are given by maps $f : A^* \otimes A \to B^* \otimes B$ in $\mathbb{C}$ which are *completely positive.* Note that REL and FDHILB are biproduct dagger compact closed categories.

From this, the category CPM($\mathbb{C}$)$^\oplus$, the free biproduct completion of CPM($\mathbb{C}$) is formed, which is suitable for describing quantum computation semantics. For example, given FDHILB as our starting point, the tensor unit I is the field of complex numbers. The type of **qubit** (in FDHILB and by lifting, in CPM(FDHILB)$^\oplus$) is given as $I \oplus I$. At this stage, the necessity of the CPM construction to model physical reality can be seen in the following as in FDHILB, the morphisms initialization of a qubit: $init : I \oplus I \to$ **qubit** and destructive measure:

$meas : \mathbf{qubit} \to I \oplus I$ are inverses. However, in $\text{CPM}(\textsc{FdHilb})^{\oplus}$, these same maps are given as

$$\mathbf{qubit}^* \otimes \mathbf{qubit} \xrightarrow{meas} I \oplus I \xrightarrow{init} \mathbf{qubit}^* \otimes \mathbf{qubit}$$

by the formulae:

$$meas \begin{pmatrix} a & b \\ c & d \end{pmatrix} = (a, d), \qquad init(a, d) = \begin{pmatrix} a & 0 \\ 0 & d \end{pmatrix}.$$

Therefore, the maps are not inverses and reflect the physical reality.

Example 3.2.23 (Commutative Frobenius algebras)**.** Let $\mathbb{X}$ be a symmetric monoidal category and form $\text{CFrob}(\mathbb{X})$ as follows:

Objects: Commutative Frobenius algebras[13]: A quintuple $(X, \nabla, \eta, \Delta, \epsilon)$ where X is a k-algebra for some field k, and $\nabla : A \otimes A \to A$, $\eta : k \to A$, $\Delta : A \to A \otimes A$, $\epsilon : A \to k$ are natural maps in the algebra. Additionally, these satisfy

together with the additional property that $\Delta\nabla = 1$.

Maps: Multiplication (∇) and co-multiplication (Δ) preserving homomorphisms which do not necessarily preserve the unit.

Theorem 3.2.24. *When $\mathbb{X}$ is a symmetric monoidal category, CFrob($\mathbb{X}$) is a discrete inverse category.*

Proof. For $f : X \to Y$, define $f^{(-1)}$ as

$$Y \xrightarrow{1 \otimes \eta} Y \otimes X \xrightarrow{1 \otimes \Delta} Y \otimes X \otimes X \xrightarrow{1 \otimes f \otimes 1} Y \otimes Y \otimes X \xrightarrow{\nabla \otimes 1} Y \otimes X \xrightarrow{\epsilon \otimes 1} X$$

Using a result from [19], we need only show:

$$\left(f^{(-1)}\right)^{(-1)} = f$$

$$ff^{(-1)}f = f$$

$$ff^{(-1)}gg^{(-1)} = gg^{(-1)}ff^{(-1)}$$

We also use the following two identities from [13]:

$$(1 \otimes \eta)\nabla = id \tag{3.11}$$

$$\Delta(1 \otimes \epsilon) = id. \tag{3.12}$$

$$f^{(-1)(-1)} = (1 \otimes \eta)(1 \otimes \Delta)(1 \otimes (f^{(-1)}) \otimes 1)(\nabla \otimes 1)(\epsilon \otimes 1)$$

$$= (1 \otimes \eta)(1 \otimes \Delta)(1 \otimes ((1 \otimes \eta)(1 \otimes \Delta)(1 \otimes f \otimes 1)(\nabla \otimes 1)(\epsilon \otimes 1)) \otimes 1)(\nabla \otimes 1)(\epsilon \otimes 1)$$

$$= (1 \otimes \eta)(1 \otimes \Delta)(1 \otimes 1 \otimes \eta)(1 \otimes 1 \otimes f \otimes 1 \otimes 1)(1 \otimes \nabla \otimes 1 \otimes 1)$$

$$(1 \otimes \epsilon \otimes 1 \otimes 1)(\nabla \otimes 1)(\epsilon \otimes 1)$$

$$= (\eta \otimes 1)(\Delta \otimes 1)(1 \otimes \nabla)(f \otimes 1)(((\eta)(\Delta \otimes 1)(1 \otimes \nabla)(1 \otimes \epsilon)) \otimes 1)((1 \otimes \epsilon)$$

$$= (1 \otimes \eta)\nabla\Delta(1 \otimes \epsilon)f(\eta \otimes 1)\nabla\Delta(1 \otimes \epsilon)$$

$$= id_x id_x \ f \ id_y id_y$$

$$= f$$

$$ff^{(-1)}f = f(1 \otimes \eta)(1 \otimes \Delta)(1 \otimes f \otimes 1)(\nabla \otimes 1)(\epsilon \otimes 1)f$$

$$= (1 \otimes \eta)(1 \otimes \Delta)(f \otimes f \otimes 1)(\nabla \otimes 1)(1 \otimes f)(\epsilon \otimes 1)$$

$$= (1 \otimes \eta)(1 \otimes \Delta)(\nabla \otimes 1)(f \otimes f)(\epsilon \otimes 1)$$

$$= (1 \otimes \eta)\nabla\Delta(f \otimes f)(\epsilon \otimes 1)$$

$$= \Delta(f \otimes f)(\epsilon \otimes 1)$$

$$= f\Delta(\epsilon \otimes 1)$$

$$= f$$

Finally, to show $ff^{(-1)}$ and $gg^{(-1)}$ commute:

$$f(1 \otimes \eta)(1 \otimes \Delta)(1 \otimes f \otimes 1)(\nabla \otimes 1)(\epsilon \otimes 1)g(1 \otimes \eta)(1 \otimes \Delta)(1 \otimes g \otimes 1)(\nabla \otimes 1)(\epsilon \otimes 1)$$

$$= (1 \otimes \eta)(1 \otimes \Delta)(\nabla \otimes 1)(f \otimes 1)(\epsilon \otimes 1)(1 \otimes \eta)(1 \otimes \Delta)(\nabla \otimes 1)(g \otimes 1)(\epsilon \otimes 1)$$

$$= (1 \otimes \eta)\nabla\Delta(f \otimes 1)(\epsilon \otimes 1)(1 \otimes \eta)\nabla\Delta(g \otimes 1)(\epsilon \otimes 1)$$

$$= \Delta(f \otimes 1)(\epsilon \otimes 1)\Delta(g \otimes 1)(\epsilon \otimes 1)$$

$$= \Delta(1 \otimes \Delta)(f \otimes g \otimes 1)(\epsilon \otimes \epsilon \otimes 1)$$

$$= \Delta(1 \otimes \Delta)(g \otimes f \otimes 1)(\epsilon \otimes \epsilon \otimes 1) \qquad\qquad \text{co-commutativity}$$

$$= gg^{(-1)}ff^{(-1)}$$

$\square$

Chapter 4

Quantum computation and circuits

4.1 Linear algebra

Quantum computation requires familiarity with the basics of linear algebra. This section will give definitions of the terms used throughout this thesis.

4.1.1 Basic definitions

The first definition needed is that of a *vector space*.

Definition 4.1.1 (Vector Space). Given a field F, whose elements will be referred to as scalars, a *vector space* over F is a non-empty set V with two operations, *vector addition* and *scalar multiplication*. *Vector addition* is defined as $+ : V \times V \to V$ and denoted as $\mathbf{v} + \mathbf{w}$ where $\mathbf{v}, \mathbf{w} \in V$. The set V must be an abelian group under $+$. *Scalar multiplication* is defined as $: F \times V \to V$ and denoted as $c\mathbf{v}$ where $c \in F, \mathbf{v} \in V$. Scalar multiplication distributes over both vector addition and scalar addition and is associative. F's multiplicative identity is an identity for scalar multiplication.

The specific algebraic requirements are:

1. $\forall \mathbf{u}, \mathbf{v}, \mathbf{w} \in V, \ (\mathbf{u} + \mathbf{v}) + \mathbf{w} = \mathbf{u} + (\mathbf{v} + \mathbf{w});$

2. $\forall \mathbf{u}, \mathbf{v} \in V, \ \mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u};$

3. $\exists \mathbf{0} \in V$ such that $\forall \mathbf{v} \in V, \mathbf{0} + \mathbf{v} = \mathbf{v};$

4. $\forall \mathbf{u} \in V, \exists \mathbf{v} \in V$ such that $\mathbf{u} + \mathbf{v} = \mathbf{0};$

5. $\forall \mathbf{u}, \mathbf{v} \in V, c \in F, \ c(\mathbf{u} + \mathbf{v}) = c\mathbf{u} + c\mathbf{v};$

6. $\forall \mathbf{u} \in V, c, d \in F, \ (c + d)\mathbf{u} = c\mathbf{u} + d\mathbf{u}$;

7. $\forall \mathbf{u} \in V, c, d \in F, \ (cd)\mathbf{u} = c(d\mathbf{u})$;

8. $\forall \mathbf{u} \in V, \ 1\mathbf{u} = \mathbf{u}$.

Examples of vector spaces over F are: $F^{n \times m}$ – the set of $n \times m$ matrices over F; and F^n – the $n-$fold Cartesian product of F. $F^{n \times 1}$, the set of $n \times 1$ matrices over F is also called the space of column vectors, while $F^{1 \times n}$, the set of row vectors. Often, F^n is identified with $F^{n \times 1}$.

This thesis shall identify F^n with the column vector space over F.

Definition 4.1.2 (Linearly independent). A subset of vectors $\{\mathbf{v}_i\}$ of the vector space V is said to be *linearly independent* when no finite linear combination of them, $\sum a_j \mathbf{v}_j$ equals $\mathbf{0}$ unless all the a_j are zero.

Definition 4.1.3 (Basis). A *basis* of a vector space V is a linearly independent subset of V that generates V. That is, any vector $u \in V$ is a linear combination of the basis vectors.

4.1.2 Matrices

As mentioned above, the set of $n \times m$ matrices over a field is a vector space. Additionally, matrices compose and the tensor product of matrices is defined.

Matrix composition is defined as usual. That is, for $A = [a_{ij}] \in F^{m \times n}, B = [b_{jk}] \in F^{n \times p}$:

$$A B = \left[\left(\sum_j a_{ij} b_{jk} \right)_{ik} \right] \in F^{m \times p}.$$

Definition 4.1.4 (Diagonal matrix). A *diagonal matrix* is a matrix where the only non-zero entries are those where the column index equals the row index.

The diagonal matrix $n \times n$ with only 1's on the diagonal is the identity for matrix multiplication, and is designated by I_n.

Definition 4.1.5 (Transpose)**.** The *transpose* of an $n \times m$ matrix $A = [a_{ij}]$ is an $m \times n$ matrix A^t with the i, j entry being a_{ji}.

When the base field of a matrix is $\mathbb{C}$, the complex numbers, the *conjugate transpose* (also called the *adjoint*) of an $n \times m$ matrix $A = [a_{ij}]$ is defined as the $m \times n$ matrix A^* with the i, j entry being $\bar{a}_{ji}$, where $\bar{a}$ is the complex conjugate of $a \in \mathbb{C}$.

When working with column vectors over $\mathbb{C}$, note that $\mathbf{u} \in \mathbb{C}^n \implies \mathbf{u}^* \in \mathbb{C}^{1 \times n}$ and that $\mathbf{u}^* \times \mathbf{u} \in \mathbb{C}^{1 \times 1}$. This thesis will use the usual identification of $\mathbb{C}$ with $\mathbb{C}^{1 \times 1}$. A column vector $\mathbf{u}$ is called a *unit vector* when $\mathbf{u}^* \times \mathbf{u} = 1$.

Definition 4.1.6 (Trace)**.** The *trace*, $Tr(A)$ of a square matrix $A = [a_{ij}]$ is $\sum a_{ii}$.

Tensor Product

The tensor product of two matrices is the usual Kronecker product:

$$
U \otimes V =
\begin{bmatrix}
u_{11}V & u_{12}V & \cdots & u_{1m}V \\
u_{21}V & u_{22}V & \cdots & u_{2m}V \\
\vdots & \vdots & \ddots & \\
u_{n1}V & u_{n2}V & \cdots & u_{nm}V
\end{bmatrix}
=
\begin{bmatrix}
u_{11}v_{11} & \cdots & u_{12}v_{11} & \cdots & u_{1m}v_{1q} \\
u_{11}v_{21} & \cdots & u_{12}v_{21} & \cdots & u_{1m}v_{2q} \\
\vdots & \vdots & \vdots & \ddots \\
u_{n1}v_{p1} & \cdots & u_{n2}v_{p1} & \cdots & u_{nm}v_{pq}
\end{bmatrix}
$$

Special matrices

When working with quantum values certain types of matrices over the complex numbers are of special interest. These are:

Unitary Matrix : Any $n \times n$ matrix A with $AA^* = I \ (= A^*A)$.

Hermitian Matrix : Any $n \times n$ matrix A with $A = A^*$.

Positive Matrix : Any Hermitian matrix A in $\mathbb{C}^{n \times n}$ where $\mathbf{u}^*A\mathbf{u} \geq 0$ for all vectors $\mathbf{u} \in \mathbb{C}^n$.
Note that for any Hermitian matrix A and vector u, $\mathbf{u}^*A\mathbf{u}$ is real.

Completely Positive Matrix : Any positive matrix A in $\mathbb{C}^{n \times n}$ where $I_m \otimes A$ is positive.

The matrix

$$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

is an example of a matrix that is *unitary*, *Hermitian*, *positive* and *completely positive*.

Superoperators

A *Superoperator* S is a matrix over $\mathbb{C}$ with the following restrictions:

1. S is *completely positive*. This implies that S is positive as well.

2. For all positive matrices A, $Tr(S\,A) \leq Tr(A)$.

4.2   Basic quantum computation

4.2.1   Quantum bits

Quantum computation deals with operations on **qubit**s. A **qubit** is typically represented in the literature on quantum computation as a complex linear combination of $|0\rangle$ and $|1\rangle$, respectively identified with (1,0) and (0,1) in $\mathbb{C}^2$. Because of the identification of the basis vectors, any **qubit** can be identified with a non-zero vector in $\mathbb{C}^2$. In standard quantum computation, the important piece of information in a **qubit** is its direction rather than amplitude. In other words, given $q = \alpha\,|0\rangle + \beta\,|1\rangle$ and $q' = \alpha'\,|0\rangle + \beta'\,|1\rangle$ where $\alpha = \gamma\alpha'$ and $\beta = \gamma\beta'$, then q and q' represent the same quantum state.

A **qubit** that has either α or β zero is said to be in a *classical state*. Any other combination of values is said to be a *superposition*.

will introduce quantum circuits which act on **qubit**s. This section will have some forward references to circuits to illustrate points introduced here.

4.2.2 Quantum entanglement

Consider what happens when working with a pair of **qubit**s, p and q. This can be considered as the a vector in $\mathbb{C}^4$ and written as

$$\alpha_{00}\left|00\right\rangle + \alpha_{01}\left|01\right\rangle + \alpha_{10}\left|10\right\rangle + \alpha_{11}\left|11\right\rangle. \tag{4.1}$$

In the case where p and q are two independent **qubit**s, with $p = \alpha\left|0\right\rangle + \beta\left|1\right\rangle$ and $q = \gamma\left|0\right\rangle + \delta\left|1\right\rangle$,

$$p \otimes q = \alpha\gamma\left|00\right\rangle + \alpha\delta\left|01\right\rangle + \beta\gamma\left|10\right\rangle + \beta\delta\left|11\right\rangle \tag{4.2}$$

where $p \otimes q$ is the standard tensor product of p and q regarded as vectors. There are states of two **qubit**s that cannot be written as a tensor product. As an example, the state

$$\frac{1}{\sqrt{2}}\left|00\right\rangle + \frac{1}{\sqrt{2}}\left|11\right\rangle \tag{4.3}$$

is not a tensor product of two distinct **qubit**s. In this case the two **qubit**s are said to be *entangled*.

4.2.3 Quantum gates

Quantum gates operate on **qubit**s. These gates are conceptually similar to logic gates in the classical world. In the classical world the only non-trivial single **bit** gate is the Not gate which sends 0 to 1 and 1 to 0. However, there are infinitely many non-trivial quantum gates.

An $n-$**qubit** quantum gate is represented by a $2^n \times 2^n$ matrix. A necessary and sufficient condition for such a matrix to be a quantum gate is that it is *unitary*.

The entanglement of two **qubit**s, p and q, is accomplished by applying a Hadamard transformation to p followed by a Not applied to q controlled by p. The circuit in figure 4.2 on page 79 shows how to entangle two **qubit**s that start with an initial state of $\left|00\right\rangle$. See this can be done in L-QPL.

A list of some common gates, together with their usual quantum circuit representation is given in the next section in table 4.1 on page 80.

4.2.4 Measurement

The other allowed operation on a **qubit** or group of **qubit**s is measurement. When a **qubit** is measured it assumes only one of two possible values, either $|0\rangle$ or $|1\rangle$. Given

$$q = \alpha \, |0\rangle + \beta \, |1\rangle \tag{4.4}$$

where $|\alpha|^2 + |\beta|^2 = 1$, then measuring q will result in $|0\rangle$ with probability $|\alpha|^2$ and $|1\rangle$ with probability $|\beta|^2$. Once a **qubit** is measured, re-measuring will always produce the same value.

In multi-**qubit** systems the order of measurement does not matter. If p and q are as in equation (4.1) on the following page, let us suppose measuring p gives $|0\rangle$. The measure will result in that value with probability $|\alpha_{00}|^2 + |\alpha_{01}|^2$, after which the system collapses to the state:

$$\alpha_{00} \, |00\rangle + \alpha_{01} \, |01\rangle \tag{4.5}$$

Measuring the second **qubit**, q, will give $|0\rangle$ with probability $|\alpha_{00}|^2$ or $|1\rangle$ with probability $|\alpha_{01}|^2$.

Conversely, if q was measured first and gave us $|0\rangle$ (with a probability of $|\alpha_{00}|^2 + |\alpha_{10}|^2$) and then p was measured, p will give us $|0\rangle$ with probability $|\alpha_{00}|^2$ or $|1\rangle$ with probability $|\alpha_{10}|^2$.

Thus, when measuring both p and q, the probability of getting $|0\rangle$ from both measures is $|\alpha_{00}|^2$, regardless of which **qubit** is measured first.

Considering states such as in equation (4.3), measuring either **qubit** would actually force the other **qubit** to the same value. This type of entanglement is used in many quantum algorithms such as quantum teleportation.

4.2.5 Mixed states

The notion of *mixed states* refers to an outside observer's knowledge of the state of a quantum system. Consider a 1 **qubit** system

$$\nu = \alpha \left|0\right\rangle + \beta \left|1\right\rangle. \tag{4.6}$$

If ν is measured but the results of the measurement are not examined, the state of the system is either $\left|0\right\rangle$ or $\left|1\right\rangle$ and is no longer in a superposition. This type of state is written as:

$$\nu = |\alpha|^2 \{\left|0\right\rangle\} + |\beta|^2 \{\left|1\right\rangle\}. \tag{4.7}$$

An external (to the state) observer knows that the state of ν is as expressed in equation (4.7) on page 77. Since the results of the measurement were not examined, the exact state (0 or 1) is unknown. Instead, a probability is assigned as expressed in the equation. Thus, if the **qubit** ν is measured and the results are not examined, ν can be treated as a probabilistic **bit** rather than a **qubit**.

4.2.6 Density matrix notation

The state of any quantum system of **qubit**s may be represented via a *density matrix*. In this notation, given a **qubit** ν, the coefficients of $\left|0\right\rangle$ and $\left|1\right\rangle$ form a column vector u. Then the density matrix corresponding to ν is uu^*. If $\nu = \alpha \left|0\right\rangle + \beta \left|1\right\rangle$,

$$\nu = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \begin{pmatrix} \overline{\alpha} & \overline{\beta} \end{pmatrix} = \begin{pmatrix} \alpha\overline{\alpha} & \alpha\overline{\beta} \\ \beta\overline{\alpha} & \beta\overline{\beta} \end{pmatrix}. \tag{4.8}$$

When working with mixed states the density matrix of each component of the mixed state is added. For example, the mixed state shown in equation (4.7) on the next page would be represented by the density matrix

$$|\alpha|^2 \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + |\beta|^2 \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} |\alpha|^2 & 0 \\ 0 & |\beta|^2 \end{pmatrix}. \tag{4.9}$$

Note that since the density matrix of mixed states is a linear combination of other density matrices, it is possible to have two different mixed states represented by the same density matrix.

The advantage of this notation is that it becomes much more compact for mixed state systems. Additionally, scaling issues are handled by insisting the density matrix has a trace $= 1$. During a general quantum computation, as we shall see, the trace can actually fall below 1 indicating that the computation is not everywhere total.

4.2.7 Gates and density matrices

When considering a **qubit** q as a column vector and a unitary transform T as a matrix, the result of applying the transform T to q is the new vector Tq. The density matrix of the original **qubit** is given by $q\,q^*$, while the density matrix of the transformed **qubit** is $(Tq)(Tq)^*$, which equals $T(qq^*)T^*$. Thus, when a **qubit** q is represented by a density matrix A, the formula for applying the transform T to q is TAT^*.

4.3   Quantum circuits

4.3.1   Contents of quantum circuits

Currently a majority of quantum algorithms are defined and documented using *quantum circuits.* These are wire-type diagrams with a series of **qubit**s input on the left of the diagram and output on the right. Various graphical elements are used to describe quantum gates, measurement, control and classical **bit**s.

Gates and **qubit**s

The simplest circuit is a single wire with no action:

$$\underline{\quad x \quad}$$

The next simplest circuit is one **qubit** and one gate. The **qubit** is represented by a single wire, while the gate is represented by a box with its name, G, inside it. This is shown in the circuit in figure 4.1 on page 79. In general, the name of the wire which is input to the gate G may be different from the name of G's output wire. Circuit diagrams may also contain constant components as input to gates as in the circuit in figure 4.3 on page 79.
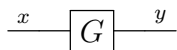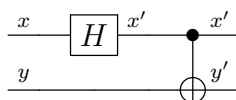

Figure 4.1: Simple single gate circuit
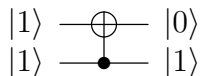

Figure 4.2: Entangling two **qubit**s.

Figure 4.3: Controlled-Not of $|1\rangle$ and $|1\rangle$

Future diagrams will drop the wire labels except when they are important to the concept under discussion.

Controlled gates, where the gate action depends upon another **qubit**, are shown by attaching a wire between the wire of the control **qubit** and the controlled gate. The circuit in figure 4.2 on page 79 shows two **qubit**s, where a Hadamard is applied to the top **qubit**, followed by a Controlled-Not applied to the second **qubit**. In circuits, the control **qubit** is on the vertical wire with the solid dot. This is then connected via a horizontal wire to the gate being controlled.

A list of common gates, their circuits and corresponding matrices is given in table 4.1 on the next page.

Measurement

Measurement is used to transform the quantum data to classical data so that it may be then used in classical computing (e.g. for output). The act of measurement is placed at the last
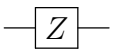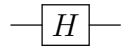
Gate	Circuit	Matrix
Not (X)		$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Z		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard		$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Swap		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Controlled-Not		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Toffoli		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

Table 4.1: Gates, circuit notation and matrices

part of the quantum algorithm in many circuit diagrams and is sometimes just implicitly considered to be there.

While there are multiple notations used for measurement in quantum circuit diagrams, this thesis will standardize on the *D-box* style of measurement as shown in .
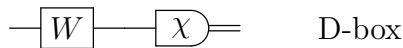
Figure 4.4: Measure notation in quantum circuits

A measurement may have a double line leaving it, signifying a **bit**, or nothing, signifying a destructive measurement.

Operations affecting multiple **qubit**s at the same time are shown by extending the gate or measure box to encompass all desired wires. In the circuit in , the gate U applies to all of the first three **qubit**s and the measurement applies to the first two **qubit**s.
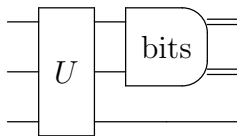
Figure 4.5: Examples of multi-**qubit** gates and measures

0-control and control by **bit**s

The examples above have only shown control based upon a **qubit** being $|1\rangle$. Circuits also allow control on a **qubit** being $|0\rangle$ and upon classical values. These forms of control are illustrated by the circuit in with four **qubit**s (r_1, r_2, p and q).

At g_1, a Hadamard is $1-$controlled by r_2 and is applied to each of r_1 and p. This is followed in column g_2 with the Not transform applied to r_2 being $0-$controlled by r_1 . In the same column, a Z gate is $0-$controlled by q and applied to p. p and q are then measured in column g_3 and their corresponding classical values are used for control in g_4. In g_4, the

U_R gate is applied to both r_1 and r_2, but only when the measure result of p is 0 and the measure result of q is 1.
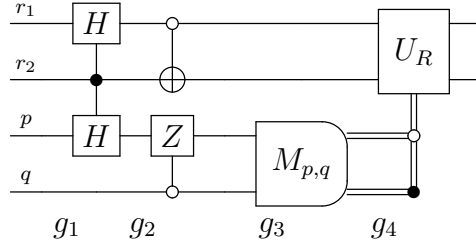
Figure 4.6: Other forms of control for gates

Multi-**qubit** lines

It is common to represent multiple **qubit**s on one line. A gate applied to a multi-**qubit** line must be a tensor product of gates of the correct dimensions. The circuit in figure 4.7 on the next page shows n **qubit**s on one line with the Hadamard gate (tensored with itself n times) applied to all of them. That is followed by a unary gate U_R tensored with $I^{\otimes(n-2)}$ and tensored with itself again. This will have the effect of applying an U_R gate to the first and last **qubit**s on the line.

Figure 4.7: n **qubit**s on one line

Other common circuit symbols

Two other symbols that are regularly used are the swap and controlled-$\mathbb{Z}$, shown in the circuit in figure 4.8 on the following page. Note that swap is just shorthand for a series of three controlled-Not gates with the control **qubit** changing. This can also be seen directly by multiplying the matrices for the controlled-Not gates as shown in equation (4.10) on page 83.

$$
\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{4.10}
$$
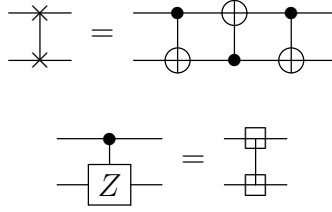
Figure 4.8: Swap and controlled-Z

4.3.2 Syntax of quantum circuits

Quantum circuits were originally introduced by David Deutsch in [10]. He extended the idea of standard classical based gate diagrams to encompass the quantum cases. In his paper, he introduced the concepts of quantum gates, sources of **bit**s, sinks and universal gates. One interesting point of the original definition is that it *does* allow loops. Currently, the general practice is not to allow loops of **qubit**s. The commonly used elements of a circuit are summarized in table 4.2 on page 84.

A valid quantum circuit must follow certain restrictions. As physics requires **qubit**s must not be duplicated, circuits must enforce this rule. Therefore, three restrictions in circuits are the *no fan-out*, *no fan-in* and *no loops* rules. These conditions are a way to express the *linearity* of quantum algorithms. Variables (wires) may not be duplicated, may not be destroyed without a specific operation and may not be amalgamated.

4.3.3 Examples of quantum circuits

This section will present three quantum algorithms and the associated circuits. Each of these circuits presented may be found in [17].

First, *quantum teleportation*, an algorithm which sends a quantum bit across a distance via the exchange of two classical bits. This is followed by the *Deutsch-Jozsa algorithm*, which provides information about the global nature of a function with less work than a classical deterministic algorithm can. The third example is circuits for the *quantum Fourier transformation* and its inverse.
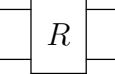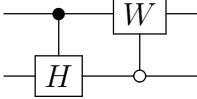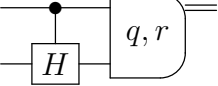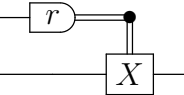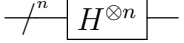
Desired element	Element in a quantum circuit diagram.	Example
qubit	A single horizontal line.	—
Classical **bit**	A double horizontal line.	=
Single-**qubit** gates	A box with the gate name (G) inside it, one wire attached on its left and one wire attached on the right.	
Multi-**qubit** gates	A box with the gate name (R) inside it, n wires on the left side and the same number of wires on the right.	
Controlled **qubit** gates	A box with the gate name (H, W) inside, with a solid (1-control) or open (0-control) dot on the control wire with a vertical wire between the dot and the second gate.	
Controlled-Not gates	A *target* $\oplus$, with a solid (1-control) or open (0-control) dot on the control wire with a vertical wire between the dot and the gate.	
Measurement	A D shaped node with optional names or comments inside. One to n single wires are attached on the left (**qubit**s coming in) and 0 to m classical **bit** wires on the right where $m \leq n$. Classical **bit**s may be dropped as desired.	
Classical control	Control bullets are attached to horizontal classical wires, with vertical classical wires attached to the controlled gate.	
Multiple **qubit**s	Annotate the line with the number of **qubit**s and use tensors on gates.	

Table 4.2: Syntactic elements of quantum circuit diagrams

Quantum teleportation

The standard presentation of this algorithm involves two participants A and B. (Henceforth known as Alice and Bob). Alice and Bob first initialize two **qubit**s to $|00\rangle$, then place them into what is known as an *EPR* (for Einstein, Podolsky and Rosen) state. This is accomplished by first applying the Hadamard gate to Alice's **qubit**, followed by a Controlled-Not to the pair of **qubit**s controlled by Alice's **qubit**.

Then, Bob travels somewhere distant from Alice, taking his **qubit** with him[1].

At this point, Alice receives a **qubit**, ν, in an unknown state and has to pass ν on to Bob. She then uses ν as the control and applies a Controlled-Not transform to this new pair. Alice then applies a Hadamard transform applied to ν.

Alice now measures the two **qubit**s and sends the resulting two **bit**s of classical information to Bob.

Bob then examines the two **bit**s that he receives from Alice. If the **bit** resulting from measuring Alice's original bit is 1, he applies the Not (also referred to as X) gate $\left(= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\right)$ to his **qubit**. If the measurement result of ν is one, he applies the Z gate $\left(= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}\right)$. Bob's **qubit** is now in the same state as the **qubit** Alice wanted to send. The circuit for this is shown in .
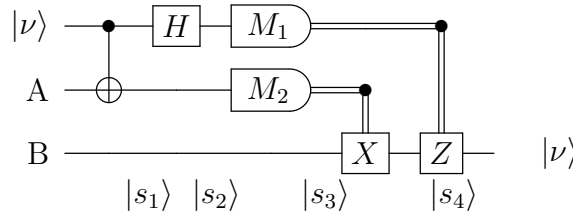
Figure 4.9: Quantum teleportation

For comparison , see showing how this would be implemented in L-QPL.

---

[1]Notice that all other physical constraints are ignored in this algorithm. There is no concern about how one separates the **qubit**s, transports the **qubit** or potential decoherence of the **qubit**s.

Deutsch-Jozsa algorithm

The Deutsch-Jozsa algorithm describes a way of determining whether a function f[2] is *constant* (i.e. always 0 or 1) or *balanced* (i.e. produces an equal number of 0 or 1 results) based on applying it to one quantum bit. The function takes n **bit**s as input and produces a single **bit**.

f is assumed to be an expensive function, therefore, a desired effect is to evaluate f as few times as possible before determining if f is balanced or constant. The worst case scenario when evaluating f classically is that determining the result requires $2^{n-1} + 1$ invocations of the function. The best possible case is 2 invocations, which occurs when f is balanced and the first two inputs chosen produce different results.

The quantum circuit requires only one application of the function to $n+1$ **qubit**s which have been suitably prepared to make the decision.

The algorithm relies on being able to construct an $n + 1$ order unitary operator based upon f. In general, a unitary operator like this may be constructed by mapping the state $|a, b\rangle$ to $|a, b \oplus f(a)\rangle$ where $\oplus$ is the exclusive-or operator and a is n **bit** values. If we name this operator U_f, the circuit in will solve the problem with just one application. See the appendix, for how this would be done in L-QPL.
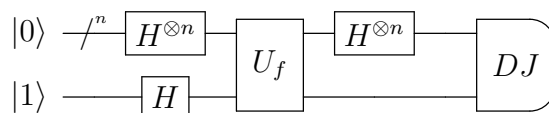
Figure 4.10: Circuit for the Deutsch-Jozsa algorithm

The idea of quantum parallelism is what makes this and many other quantum algorithms work. The initial state of the system is set to $|0^{\otimes n} \otimes 1\rangle$ after which the Hadamard gate is applied to all of the **qubit**s. This places the input **qubit**s into a superposition of all possible input values and the answer **qubit** is a superposition of 0 and 1. At this point, the unitary

---

[2]The obvious pre-condition for the Deutsch-Jozsa algorithm is that the function f is *either* balanced or constant and not some general function. The results are not well-defined if f does not fit into one of the two possible categories.

transformation U_f is applied to the **qubit**s. Then the Hadamard transform is applied again to the input **qubit**s.

To complete the algorithm, measure *all* the **qubit**s. It can be shown that if f is constant, the input **qubit**s will all measure to 0, while if it is balanced, at least one of those **qubit**s will be 1.

Quantum Fourier transform

The circuits for the quantum Fourier transformation and its inverse are in figure 4.11 on page 87 and figure 4.12 on page 88 respectively. These transforms are used extensively in many quantum algorithms, including Shor's factoring algorithm.

The quantum Fourier transform is definable on an arbitrary number of **qubit**s. This is typically presented by eliding the 3rd to the $n-3^{\text{rd}}$ lines and interior processing. The L-QPL code for the quantum Fourier transform is in the appendix, In this circuit, the parametrized transform R_n is the rotation transform, given by:

$$R_n = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi i}{2^n}} \end{bmatrix}$$
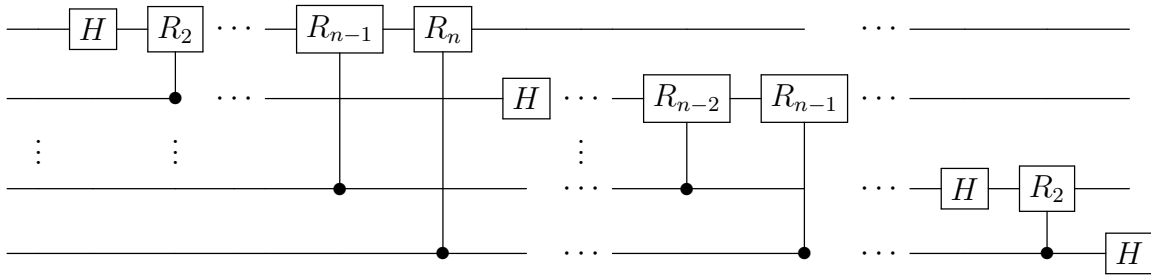
Figure 4.11: Circuit for the quantum Fourier transform

The inverse of a circuit is determined by traversing the circuit from right to left. This process changes the original quantum Fourier circuit to its inverse as shown in figure 4.12 on page 88. The L-QPL code for the inverse quantum Fourier transform is in the appendix,
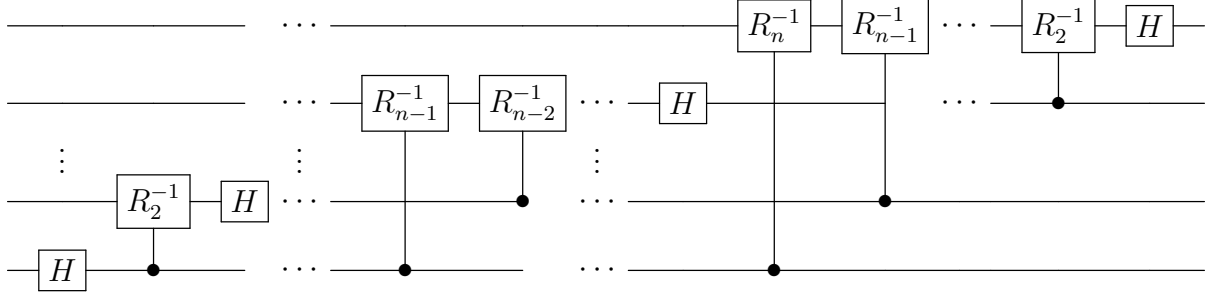
Figure 4.12: Circuit for the inverse quantum Fourier transform

4.4 Extensions to quantum circuits

To facilitate the transition to the programming language L-QPL, this section introduces three extensions to quantum circuits. The extensions are *renaming, wire bending and crossing,* and *scoped control*. Each extension adds expressive power to quantum circuits but does not change the semantic power. For each of the extensions, examples of how to re-write the extension in standard quantum circuit terminology will be provided.

4.4.1 Renaming

Quantum circuits currently allow renaming to be an implicit part of any gate. The circuit in figure 4.13 on the following page gives an operation to explicitly do this and its rewriting in standard circuit notation.

Figure 4.13: Renaming of a **qubit** and its equivalent diagram

4.4.2 Wire crossing

Crossing and bending of wires in a circuit diagram is added to allow a simpler presentation of algorithms. The circuit in figure 4.14 on page 89 illustrates the concept of re-organizing and bending of wires.
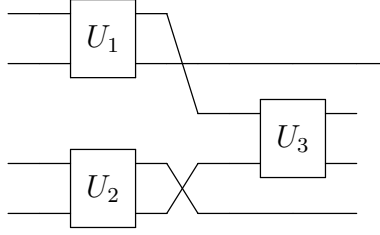
Figure 4.14: Bending

4.4.3 Scoped control

This extension allows us to group different operations in a circuit and show that all of them are controlled by a particular **qubit**. This is the same as attaching separate control wires to each of the gates in the grouped operations. Measurements are not affected by control. Figure 4.15 on page 89 shows a scoped control box on the left which includes a measure. The right hand side of the same figure shows the circuit translated back to standard circuit notation, with the measure not being affected by the control.
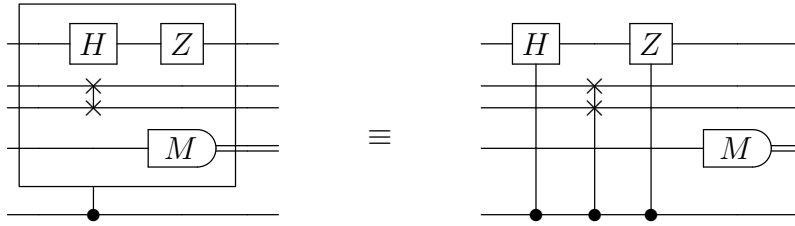
Figure 4.15: Scope of control

Scoping boxes correspond to procedures and blocks in L-QPL.

Naturally, both scoping and bending may be combined as in figure 4.16 on the next page.
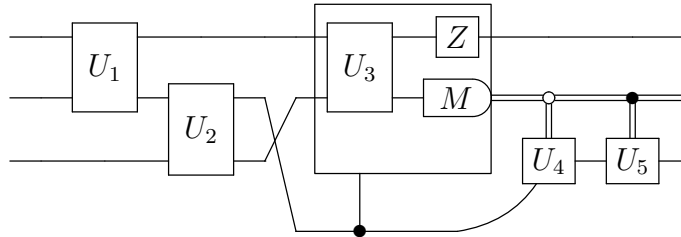
Figure 4.16: Extensions sample

However, note that exchanging wires is not the same as swap. Exchanging a pair of

wires is not affected by control, but a swap is affected by control, as shown in figure 4.17 on page 90.
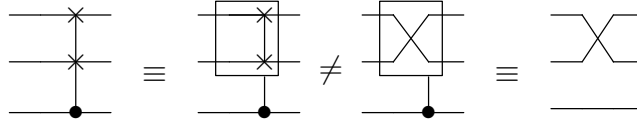
Figure 4.17: Swap in control vs. exchange in control

4.4.4 Circuit identities

Circuits allow the writing of the same algorithm in multiple ways. This sub-section will list some of the circuit identities that hold with the extended notation.

First, note that although a measure may appear inside a control box, it is not affected by the control, as in figure 4.18 on the next page. Conversely, a measurement commutes with control of a circuit as in figure 4.18 on the following page.
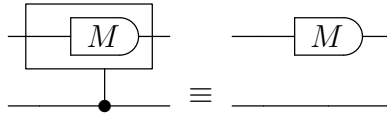
Figure 4.18: Measure is not affected by control
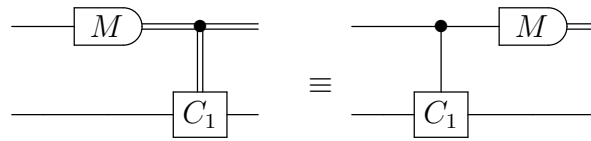
Figure 4.19: Control is not affected by measure

One of the notations introduced earlier was that of *0-control*. This type of control is the same as applying a *Not* transform before and after a *1-control*, as shown in figure 4.20 on the next page.

Figure 4.21 on the following page shows that scoped control of multiple transforms is the same as controlling those transforms individually. Figure 4.22 on the next page similarly
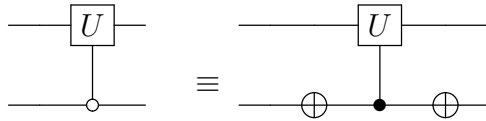
Figure 4.20: Zero control is syntactic sugar
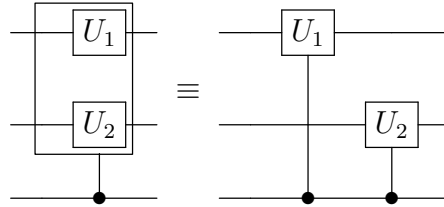
Figure 4.21: Scoped control is parallel control

shows that scoped control of multiple transforms of the same **qubit** is the same as controlling those transforms serially.
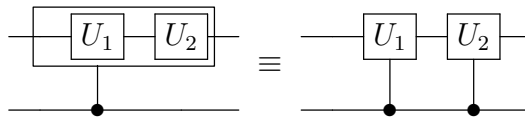
Figure 4.22: Scoped control is serial control

Multiple control commutes with scoping as shown in figure 4.23 to figure 4.24 on page 92.

4.5   An alternate description of quantum circuits

In order to explore transforms of quantum circuits, it is helpful to have a algebraic description which will allow manipulation of the circuits. The goals of the algebraic description are:

- Represent **qubit**s and **bit**s;

- represent gates;

- allow algorithmic manipulations of the circuit;

- allow proving that correctness of manipulations.

Note that measurement is not included in this representation.
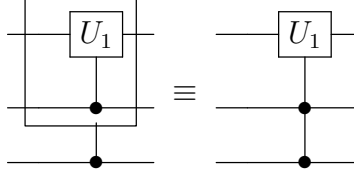
Figure 4.23: Multiple control
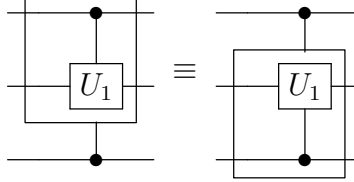
Figure 4.24: Control scopes commute

4.5.1 Base types

The base types **bit** and **qubit** are needed for any circuit. These will be taken as elemental and correspond to the classical notions of "bit" (i.e., 0 or 1) and "qubit" (i.e., $\alpha\,|0\rangle + \beta\,|1\rangle$).

Definition 4.5.1. The type **endpoint** is either **bit** or **qubit**.

Definition 4.5.2. The type $\mathbf{Q} - \mathbf{Arity}$ is a partial map from $W \subset \mathbb{N}$ to **Endpoint**.

Definition 4.5.3. A *wire* is an element of $W \subset \mathbb{N}$. A *typed wire* is a wire together with a specified $\mathbf{Q} - \mathbf{Arity}$.

Definition 4.5.4. A *control wire* is a pair (w, b) where w is a typed wire and $b :: \mathbf{Bool}$.

A quantum program uses *typed wire*s as its data.

Definition 4.5.5. A *gate* is a function from W_1, a set of typed wires, to W_2, another set of typed wires. The gate function must be a superoperator as defined in [22].

4.5.2 Types and Shapes

Although circuits, gates and low level subroutines are defined at the level of wires, programmatically, we would like to refer to groupings of wires. For example, a list of **qubit** or a register (tuple) of **qubit**. These groupings may have a required leaf type (e.g., **qubit** or

bit) or it may be considered polymorphic. Additionally, the leaf type may be mixed or homogenous.

Definition 4.5.6. The singleton type $\mathbf{B}$ is defined as having the instance $B_{()}$ and the singleton type $\mathbf{Q}$ is defined as having the singleton instance $Q_{()}$.

Definition 4.5.7. The type family $\mathbf{QCData}$ consists of algebraic data types built up from **bit** or **qubit**.

Examples of $\mathbf{QCData}$ include $(\mathbf{bit}, \mathbf{qubit}, [\mathbf{qubit}])$ and **bit**.

Definition 4.5.8. Given an instance I of a type $\mathbf{T}$ in $\mathbf{QCData}$, the *shape* of I is the instance I_s obtained from I by replacing all terms of type **bit** by $B_{()}$ and all terms of type **qubit** by $Q_{()}$.

For example, a 2 element list of **qubit**s has shape $[Q_{()}, Q_{()}]$ while a pair of a **qubit** and **bit** has shape $(Q_{()}, B_{()})$

The concepts of shape and the type $\mathbf{QCData}$ allow us to group the wires of a quantum circuit into higher order types.

Chapter 5

Transformations of Quantum Programs

5.1 Subroutines

In the following, we will assume *gate* as above is a given and that $W \subset \mathbb{N}$ is fixed and finite. We will use typing notation to show membership in W — $w \in W$ is equivalent to $w :: W$.

5.1.1 Definition of a Subroutine

The concept of *subroutine* as defined below is intended to capture the essence of a describable computation in a quantum language. The low level subroutine is considered in isolation, meaning that it contains no information regarding how it fits into a larger circuit.

Definition 5.1.1. A *bare subroutine* is defined as a list of gates, written as $[g_0, g_1, \ldots, g_n]$. The list of gates must satisfy the following:

- range $g_i = \text{dom } g_{i+1}$ for $i \in \{0, 1, \ldots, n-1\}$.

A bare subroutine B can then be viewed as a function from $\text{dom } g_0$ to range g_n by applying each gate in order to $\text{dom } g_0$.

Definition 5.1.2. A *low level subroutine* is a bare subroutine with a triple (C, I, O) where each of C, I, O are of type **Arity** and

$$\text{dom } C \cap \text{dom } I = \phi \tag{5.1}$$

$$\text{dom } C \cap \text{dom } O = \phi \tag{5.2}$$

In the definition of the low level subroutine, the triple (C, I, O) describes the inputs and outputs of the subroutine. C describes the control wires, which are inputs and outputs without change, I the input wires and O the output wires.

The above data together with three additional flags provides everything we need to know regarding a subroutine:

Definition 5.1.3. A *subroutine* is a low level subroutine together with a tripartite flag c with values in $\{N, B, Q\}$, and two boolean flags, r and n.

The three flags describe the ways this subroutine may be used. Each of these flags provide information that the calling quantum program uses to determine the ways the subroutine may be called:

- *Controllable*: When c is B, a calling program may make this subroutine the target of one or more *control wire*s with type **bit**. When c is Q, the control wires my be of type **bit** or **qubit**. When c is N, no control wires may be used. Note this is separate from the C wires of the subroutine, which may be used for internally controlling portions of the subroutine.

- *Reversible*: A calling program may specify the subroutine run normally or reversed.

- *No-controllable*: In the case where this subroutine is part of a preparation / unpreparation in a (prep, transform, unprep) sequence and that sequence is controlled, then the control wires may be ignored for this subroutine.

Noting that the domains of C and I and the domains of C and O do not overlap, we can also provide an ordering of the inputs and outputs for a low level subroutine. This ordering is used for display purposes and has no additional semantic content.

Definition 5.1.4. An *ordering* of a low level subroutine is a pair of bijections, (i, o) such that:

$$i : \operatorname{dom} C \cup \operatorname{dom} I \leftrightarrow \{0, 1, \ldots, n-1\} \tag{5.3}$$

$$o : \operatorname{dom} C \cup \operatorname{dom} O \leftrightarrow \{0, 1, \ldots, m-1\} \tag{5.4}$$

where $|\operatorname{dom} C \cup \operatorname{dom} I| = n$ and $|\operatorname{dom} C \cup \operatorname{dom} O| = m$.

5.1.2 Subroutine Calls

In this section, we describe the permissible bindings given two sets of wires, where the first set will be considered as control wires and the second as either input or output wires.

Definition 5.1.5. Given C and K are **Arity** functions over the same set of typed wires V, then f is a *permissible binding* to a set of typed wires W with **Arity** T_w when:

- $f : \operatorname{dom} C + \operatorname{dom} K \to W$,

- $\forall x, y \in \operatorname{dom} f, f(x) = f(y) \implies x = y \vee x, y \in \operatorname{dom} C$,

- $x \in \operatorname{dom} C \wedge C(x) = \mathbf{qubit} \implies T_w(f(x)) = \mathbf{bit} \vee T_w(f(x)) = \mathbf{qubit}$,

- $x \in \operatorname{dom} C \wedge C(x) = \mathbf{bit} \implies T_w(f(x)) = \mathbf{bit}$,

- $x \in \operatorname{dom} K \implies T_w(f(x)) = K(x)$.

We denote the permissible bindings to W of C and K by $F(C, K, W)$.

Definition 5.1.6. In a context of typed wires W_1, a *subroutine call*, resulting in the typed wires W_2, of the subroutine $([gates], C, I, O, r, c, n)$ is a tuple $(f, g, h, i, ncf, ctrls)$ consisting of three functions, two boolean flags and a list of control wires. The functions f, g, h must satisfy:

- $f : \operatorname{dom} C \to W_1 \cap W_2$

- $g : \operatorname{dom} I \to W_1$

- $h : \operatorname{dom} O \to W_2$

- $f + g \in F(C, I, W_1)$

- $f + h \in F(C, O, W_2)$.

The two flags must satisfy:

- $i \implies r$

- $ncf \implies n.$

The control list must satisfy:

- $\forall w_c \in ctrls, \mathrm{fst}(w_c) \in W_1 \cap W_2,$

- $N = c \implies \mathrm{length}(ctrls) = 0,$

- $B = c \implies \forall w_c \in ctrls, T_1(\mathrm{fst}(w_c)) = \mathbf{bit}.$

5.1.3 High Level Structure

Let s, t be of type of the family **QCData** and a be of shape s, b be of shape t. Further, let $A = \{qt|qt :: a, qt \text{ has shape } s\}$ and $B = \{qt|qt :: b, qt \text{ has shape } t\}$, that is, A and B are the sets of quantum terms of shape s (respectively t) and type a (respectively b).

Definition 5.1.7. A *high level structure* for a call to the subroutine $([gates], C, I, O, r, c, n)$ starting in context W_1 and ending in context W_2 is a pair of maps (i_s, o_s) with $i_s : A \to F(C, I, W_1)$ and $o_s : F(C, O, W_2) \to B$.

Definition 5.1.8. Given the data for a subroutine call as above in 5.1.6 on page 96, a *structured subroutine call* is a high level structure as in 5.1.7 on page 97 and a pair of quantum terms (a, b) such that:

- $i_s(a) = f + g$ and

- $o_s(f + h) = b.$

5.2 Subroutine Calls and Transformers

We are interested in two transformed calls of subroutines. Iteration and folding. We provide the necessary information for creating either a transformed call or first transforming the subroutine and then doing a standard call as in sub-section 5.1.2 on page 96.

5.2.1 Iteration

Iteration of subroutines means to call the same subroutine within a quantum circuit some positive number of times. Discussion points:

- Can we handle the case of zero iterations? Would this just mean doing a direct mapping of the I to the O in numerical sequence?

- Can we handle the case of negative iterations? This could mean calling the inverse of the subroutine in the case where it is reversible and then iterating.

- Does iteration affect the no-control or other flags?

- The analysis below assumes that "non-linear safety" is an important property to preserve during iteration. If we remove that requirement, iteration becomes more flexible, e.g., the bijections c_b and io_b could be replaced with a single bijection $cio_b : C \cup O \leftrightarrow C \cup I$. This would allow a **qubit** that was affected by the subroutine on one iteration to be used as the control on the next iteration. (Or is this the simple case that has already been handled?)

Definition 5.2.1. Given a subroutine as in sub-section 5.1.2 on the preceding page, and a subroutine call $(f, g, h, i, ncf, ctrls)$ an *iterated call* of the subroutine $S = ([gates], C, I, O, r, c, n)$ consists of all elements of a subroutine call excepting f plus another tuple of five elements $(f_{in}, f_{out}, c_b, io_b, i_{count})$ where:

- $f_{in} : \operatorname{dom} C \to W_1 \cap W_2$

- $f_{out} : \operatorname{dom} C \to W_1 \cap W_2$

- i_{count} is a positive integer,

- c_b is a bijection (permutation) of C to C,

- io_b is a bijection between I and O.

- $f_{in} + g \in F(C, I, W_1)$

- $f_{out} + h \in F(C, O, W_2)$

- The relation $f_{out} \circ c_b^{i_{count}} \circ f_{in}^{-1}$ is a function.

Note these requirements mean that $|I| = |O|$.

Definition 5.2.2. Given the data for an iterated call, a *structured iterated call* is a high level structure as in 5.1.7 and a pair of quantum terms (a, b) such that:

- $i_s(a) = f_{in} + g$ and

- $o_s(f_{out} + h) = b$.

From the definition, note the C wires may be permuted as desired, but the combination of the f_{in}^{-1} and the permutation must leave the wires in a state where f_{out} is well-defined. See below for an example. Note the disposition of the wires due to calling the iterated subroutine is given by:

- Control mapping: $f_{out} \circ c_b^{i_{count}} \circ f_{in}^{-1}$,

- In-out mapping: $h \circ io_b^{i_{count}} \circ g^{-1}$.

Example 5.2.3 (Single call).

For this example, we will elide the details relating to high level structure, invertability, control wires and the no-control flag.

Suppose $C = [c_1, c_2, c_3], I = [i_1, i_2]$ and $O = [o_1, o_2]$. For the first part of the example, assume we are calling S from a context of $W_1 = [w_1, w_2, w_3, w_4]$, resulting in the same context (i.e., $W_1 = W_2$). In this case, the call of S can be given by:

- $f = \{c_1 \mapsto w_1, c_2 \mapsto w_1, c_3 \mapsto w_4\}$,

- $g = \{i_1 \mapsto w_2, i_2 \mapsto w_3\}$

- $h = \{o_1 \mapsto w_3, o_2 \mapsto w_2\}$

Hence we are calling S which will use w_1 as its first two control inputs and w_4 as the third. The inputs will use w_2 and w_3, while the output will "switch" those to w_3 and w_2.

Example 5.2.4 (Iterated call).

Assume S is as above and we are using the call as above. To aid in distinguishing input and output wires of the calling circuit, we will use w' for naming the output wires, so we may write $W_1 = [w_1, w_2, w_3, w_4]$ and $W_2 = [w'_1, w'_2, w'_3, w'_4]$, noting that $w_i = w'_i$ for $i \in \{1, 2, 3, 4\}$. A call iterated 5 times of S is $(g, h, i, ncf, ctrls)$ plus the tuple $(f_{in}, f_{out}, c_b, io_b, 5)$ where

- $f_{in} = \{c_1 \mapsto w_1, c_2 \mapsto w_1, c_3 \mapsto w_4\}$,

- $f_{out} = \{c_1 \mapsto w'_4, c_2 \mapsto w'_1, c_3 \mapsto w'_1\}$,

- $c_b = (c_1, c_2, c_3)$,

- $io_b = \{o_1 \mapsto i_2, o_2 \mapsto i_1\}$

If we calculate the movement of the wires for this iterated call, we see

$$(w_1, w_4) \xrightarrow{f_{in}^{-1}} C = (w_1, w_1, w_4) \xrightarrow{(1,2,3)^5} C = (w_4, w_1, w_1) \xrightarrow{f_{out}} (w'_1(= w_1), w'_4(= w_4)) \quad (5.5)$$

$$(w_2, w_3) \xrightarrow{g^{-1}} I = (w_2, w_3) \xrightarrow{io_b^5 \circ S} O = (w_3, w_2) \xrightarrow{h} (w'_3(= w_3), w'_3(= w_3)) \quad (5.6)$$

In this above example, the choice of f_{in} and f_{out} worked correctly with c_b such that the mappings were well defined. Consider however, if f_{out} were defined as $\{c_1 \mapsto w'_1, c_2 \mapsto w'_1, c_3 \mapsto w'_4\}$. We would then be expected to map / combine w_1 and w_4 into w'_1 and duplicate w_1 into both w'_1 and w'_4.

5.2.2 Iteration transformation of a subroutine

The data required to transform a subroutine is similar to that of an iterated subroutine call.

Definition 5.2.5. The *iteration transform* of a subroutine is a function Σ_i with parameters (n, c_p, io_b) which takes the subroutine $S = (g, C, I, O, r, c, n)$ to $S'(g', C', I', O', r, c, n)$, an itere. The parameters have the following types:

$$n :: \textbf{Int}, n > 0;$$

$$c_p :: \textbf{Perm}_{\textbf{C}}, \text{Permutations of the elements of } C$$

$$io_b :: \textbf{bijection}(\textbf{I} \leftrightarrow \textbf{O})$$

The function Σ_i performs the following actions:

- Type checking:

    - Ensure c_p is a permutation of the appropriate number of wires.

    - Ensure n is a positive number greater than 0.

    - Ensure io_b is a bijection between the O and I wires.

- Create n copies of *gates*.

- Connect the O wires or copy $i - 1$ to the I wires of copy i, using the bijection io_b.

- Apply the permutation c_p to the C out wires of copy $i - 1$ and connect to the input C wires of copy I.

- Apply the permutation c_p^k to the C out wires of copy n where $c_p^{n-1+k} = Id$. Connect those wires to the C' output.

- Connect the C' input wires to the C input wires of copy 1.

- Connect the I' wires to the I wires of copy 1.

- Connect the O wires of copy n to the O' wires.

The intended effect of this transform may be seen in
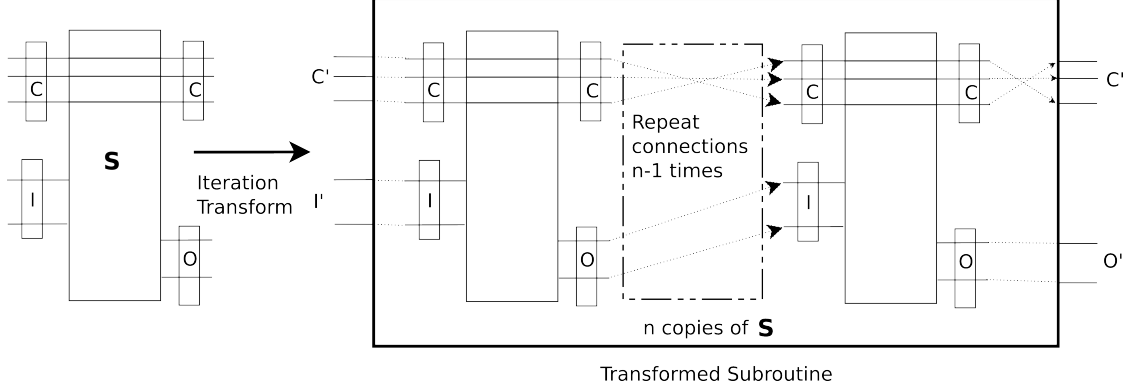
Figure 5.1: Transforming a subroutine to an iterated subroutine

5.2.3 Folding subroutines

In addition to the concept of iteration the ability to fold subroutines over data structures could also be useful. A folded call of a subroutine is similar to an iteration, in that controls and possibly some of the inputs and outputs are iterated. The difference occurs in that we expect to take some of the inputs from a data structure and return some of the outputs to a data structure. An example of this would be folding over a list of **qubit**s where each qubit is taken as a input for each iteration.

First, we will examine the requirements for a non-linear safe fold, that is, where no input duplication on the control wires is allow

Definition 5.2.6. Given a subroutine $S = ([gate], C, I, O, c, r, n)$, a starting context of typed wires W_1 and a data structure on wires $D \subset W_1$, a *linear-only folded call* of S over D resulting in the context of typed wires W_2 and the data structure $E \subset W_2$ consists of the tuple $(CI_f, CO_f, g, h, ciof_b, s_{in}, s_{out})$ where

- $CI_f :: \mathbf{Arity}, \operatorname{dom} CI_f \subset \operatorname{dom} C \cup \operatorname{dom} I$,

- $CO_f :: \mathbf{Arity}, \operatorname{dom} CO_f \subset \operatorname{dom} C \cup \operatorname{dom} O$,

- $g : \operatorname{dom} C \cup \operatorname{dom} I / \operatorname{dom} CI_f \to W_1$ and g is injective,

- $h : \operatorname{dom} C \cup \operatorname{dom} O / \operatorname{dom} CO_f \to W_2$ and h is injective,

102

- $ciof_b$ is a bijection between the sets $(\operatorname{dom} C \cup \operatorname{dom} O)/\operatorname{dom} CO_f$ and $(\operatorname{dom} C \cup \operatorname{dom} I)/\operatorname{dom} CI_f$,

- $s_{in} : \operatorname{dom} CI_f \to W_1$ (pulls from D),

- $s_{out} : \operatorname{dom} CO_f \to W_2$ (pushes to E),

- $g + s_{in} \in F(\phi, I, W_1)$,

- $ciof_b^{-1} + s_{in} \in F(\phi, I, W_1)$,

- $ciof_b + s_{out} \in F(\phi, O, W_2)$,

- $h + s_{out} \in F(\phi, O, W_2)$,

- $|\operatorname{dom} CI_f| = leafsize(D)$,

- $leafsize(E) = |\operatorname{dom} CO_f|$.

Definition 5.2.7. Given the data for an linear-only folded call, a *structured linear-only folded call* is a pair of high level structures (i_s, o_s) and (i_{fs}, o_{fs}) and a pair of quantum terms (a, b) such that:

- $i_s(a) = g + s_{in}$,

- $o_s(h + s_{out}) = b$,

- $i_{fs}(a) = s_{in}$, and

- $o_{fs}(s_{out}) = b$.

Discussion points:

- Does the structured call get rid of the last two items regarding leafsize? Same issue with non-linear safe call.

- For the structured call, it appears to me that there is only one pair of terms a, b with two different (de)structuring morphisms.

The action on the wires of the calling program will be given by this relation:

$$s_{out} + h \circ (s_{out} + cio_f + s_{in})^{len(D)} \circ (g + s_{in})^{-1}.$$

Example 5.2.8 (Fold over Carry)**.** For this example, we use the carry portion of the addition algorithm found in [25].

The carry circuit is shown below:

$$
\begin{array}{ccc}
anc & & anc \\
a & & a \\
b & & b \\
c & & c \\
\end{array}
$$
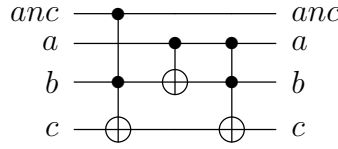
The intent of this circuit is to compute the final carry when adding the quregisters A and B. The input prepares the anc in state $|0\rangle$ and an auxiliary register R, the same size as A and B as $|00\ldots0\rangle$. Assume that $A = [w_1, w_2, w_3], B = [w_4, w_5, w_6], anc = w_7$ and $R = [w_8, w_9, w_{10}]$. (A, B, R), a tuple of three registers forms our D — the input to the fold. Then, we may perform a folded call of carry as follows:

- $\text{dom}\, C = \{anc, a\}, \ \text{dom}\, I = \{b, c\} = \text{dom}\, O;$

- $\text{dom}\, CI_f = \{a, b, c\}$ and $\text{dom}\, CO_f = \{anc, a, b\};$

- $g = \{anc \mapsto w_6\}, \ h = \{c \mapsto w_{10}\}$

- $ciof_b = \{c \mapsto anc\}$

- $s_{in} = \{a \mapsto D[0], b \mapsto D[1], c \mapsto D[2]\}$

- $s_{out} = \{anc \mapsto E[2], a \mapsto E[0], b \mapsto E[1]\}.$

From the mapping s_{out}, we set $E = (A, B, R')$ where $R' = [w_7, w_8, w_9]$.

Discussion points:

- Is there a non-linear safe use case? Carry seemed quite simple, but it required linear safe inputs when folded.

- Each fold iteration input might be multiple **qubit**s, combined into a single data structure, as in .

- The number of inputs and outputs no longer need to agree as they did in iteration. An example would be a subroutine that applied a two **qubit** gate and then discarded one of the **qubit**s. The fold would be expected to convert a list of pairs of **qubit**s to a list of **qubit**s. (Note this subroutine would not be reversible or controllable).

- The shape of the foldable in and out as well as the number of **qubit**s at each leaf would need to be known.

- The $F(C, K, W)$ permissible functions are not quite right for linear-only folds — we do not want to allow duplication of any of the inputs, so rather than $F(C, K, W)$ we must use $F(\phi, C + K, W)$.

Definition 5.2.9. Given a subroutine $S = ([gate], C, I, O, c, r, n)$, a starting context of typed wires W_1 and a data structure on wires $D \subset W_1$, a *folded call* of S over D resulting in the context of typed wires W_2 and the data structure $E \subset W_2$ consists of the tuple $(I_f, O_f, f_{in}, f_{out}, g, h, c_b, iof_b, s_{in}, s_{out})$ where

- $I_f :: \mathbf{Arity}, \mathrm{dom}\, I_f \subset \mathrm{dom}\, I,$

- $O_f :: \mathbf{Arity}, \mathrm{dom}\, O_f \subset \mathrm{dom}\, O,$

- $f_{in} : \mathrm{dom}\, C \to W_1 \cup W_2$

- $f_{out} : \mathrm{dom}\, C \to W_1 \cup W_2$

- $g : \mathrm{dom}\, I / \mathrm{dom}\, I_f \to W_1$

- $h : \mathrm{dom}\, O / \mathrm{dom}\, O_f \to W_2$

- c_b is a bijection (permutation) of C,

- iof_b is a bijection between $\mathrm{dom}\,O/\mathrm{dom}\,O_f$ and $\mathrm{dom}\,I/\mathrm{dom}\,I_f$,

- $s_{in} : \mathrm{dom}\,I_f \to W_1$ (pulls from D)

- $s_{out} : \mathrm{dom}\,O_f \to W_2$ (pushes to E)

- $f_{in} + g + s_{in} \in F(C, I, W_1)$,

- $f_{out} + h + s_{out} \in F(C, O, W_2)$,

- $iof_b + s_{in} \in F(\phi, I, W_1)$,

- $iof_b + s_{out} \in F(\phi, O, W_2)$,

- $|\mathrm{dom}\,I_f| = leafsize(D)$

- $leafsize(E) = |\mathrm{dom}\,O_f|$

Definition 5.2.10. Given the data for an folded call, a *structured folded call* is a pair of high level structures (i_s, o_s) and (i_{fs}, o_{fs}) and a pair of quantum terms (a, b) such that:

- $i_s(a) = f_{in} + g + s_{in}$,

- $o_s(f_{out} + h + s_{out}) = b$,

- $i_{fs}(a) = s_{in}$, and

- $o_{fs}(s_{out}) = b$.

5.2.4   Subroutine to folded subroutine transform

In order to transform a given subroutine, we require the following data:

- A bijection from some subset of the control and output wires to some subset of the control and input wires;

- a count of the number of iterations.

This allows us to derive a new C, I, O for the fold subroutine. We proceed with the formal definition.

Definition 5.2.11. The *fold transform* of a subroutine is a function S_f with parameters (n, b_{cio}) which takes the subroutine (g, C, I, O, r, c, n) to the subroutine $(g', C', I', O', r, c, n)$. The parameters have the following types:

$$n :: \textbf{Int}, n > 0;$$

$$b_{cio} :: \textbf{bijection}(\textbf{CI} \leftrightarrow \textbf{CO})$$

$$\text{where } CI \subseteq C \cup I \text{ and } \subseteq C \cup O.$$

Note that b_{cio} may be defined as a set of ordered pairs

$$\{(f, t) | f \in C \cup I, t \in C \cup O, \text{ and } f, t \text{ appear at most once}\}. \tag{5.7}$$

The data we need to create for the end result are the set of control wires, the input and output wires and the new gates sequence. We proceed with presenting the algorithm for the control wires.

When considering which inputs (and hence outputs) are control wires in the transformed subroutine, we must follow the path of a control input. A control input to the base subroutine will remain a control input to the transformed subroutine only if its full folded path contains only control wires before exiting. Depending upon the structure of b_{cio} it is possible all, none or some finite subset of specific control wires become controls for the transformed routine.

To determine if a wire is a control, we will calculate a characteristic of the wire and show that it requires at most k iterations to calculate, where k is the number of control wires of the original subroutine.

First, define:

$$\Omega :: C \rightarrow C \cup \{*, @\}$$

$$\Omega(c) = \begin{cases} c' & \text{if } b_{cio}(c) = c' \text{ and } c' \in C, \\ * & \text{if } c \text{ is not the first element of any pair in } b_{cio}, \\ @ & \text{if } b_{cio}(c) = j \text{ and } j \in I. \end{cases}$$

Then, noting that $k = |C|$, define:

$$\Gamma :: C \to \mathbb{N} \cup \{\infty\}$$

$$\Gamma(c) = \begin{cases} \infty & \Omega(c) = *, \\ 0 & \Omega(c) = @, \\ 1 + \Gamma(\Omega(c)) & \Gamma(\Omega(c)) < k, \\ \infty & \Gamma(\Omega(c)) \geq k. \end{cases}$$

Dually, we may define functions $\Theta(c)$ and $\Delta(c)$:

$$\Theta :: C \to C \cup \{*, @\}$$

$$\Theta(c) = \begin{cases} c' & \text{if } b_{cio}(c') = c \text{ and } c' \in C, \\ * & \text{if } c \text{ is not the second element of any pair in } b_{cio}, \\ @ & \text{if } b_{cio}(o) = c \text{ and } o \in O. \end{cases}$$

$$\Delta :: C \to \mathbb{N} \cup \{\infty\}$$

$$\Delta(c) = \begin{cases} \infty & \Theta(c) = *, \\ 0 & \Theta(c) = @, \\ 1 + \Delta(\Theta(c)) & \Delta(\Theta(c)) < k, \\ \infty & \Delta(\Theta(c)) \geq k. \end{cases}$$

Note that in the case of cycles among control wires, the cycle *must* be of size k or less. As such, at most k iterations of Γ are required before confirming a value for $\Gamma(c)$. The same argument applies to computing Θ.

Assuming that C is ordered, the data for b_{cio} may be stored such that computing $b_{cio}(c)$ and therefore $\Omega(c)$ is of $\mathcal{O}(\log k)$. Therefore, Γ is of complexity $\mathcal{O}(k \log k)$ and computing Γ for all of C will then be of $\mathcal{O}(k^2 \log k)$. Computing Δ will have the same complexity.

We may now describe the algorithm for determining control wires, C' input wires, I' and output wires, O' of the transformed subroutine. In the algorithm, n is the number of iterations, k is the cardinality of C. Additionally, we compute a *rank* of each c in C'. The *rank* of c is the number of iterations that c will go through.

1. Add all members of I to I', subscripting with 1.

2. For all $i \in I$ where $i \notin \mathrm{range}\, b_{cio}$, add $i_2, i_3, \ldots, i_n$ to I'.

3. Add all members of O to O', subscripting with n.

4. For all $o \in O$ where $o \notin \mathrm{dom}\, b_{cio}$, add $o_1, o_2, \ldots, o_{n-1}$ to O'.

5. Partition C into four subsets:

$$C_* = \{c | c \notin \mathrm{dom}\, b_{cio} \text{ and } c \notin \mathrm{range}\, b_{cio}\},$$

$$C_d = \{c | c \in \mathrm{dom}\, b_{cio} \text{ and } c \notin \mathrm{range}\, b_{cio}\},$$

$$C_r = \{c | c \notin \mathrm{dom}\, b_{cio} \text{ and } c \in \mathrm{range}\, b_{cio}\},$$

$$C_{dr} = \{c | c \in \mathrm{dom}\, b_{cio} \text{ and } c \in \mathrm{range}\, b_{cio}\}.$$

6. For each $c \in C_*$, add $c_1, c_2, \ldots, c_n$ to C'. Set the rank of each of these to 0.

7. For each $c \in C_d$, compute $j = \Gamma(c)$. If $j \geq n$, add $c_1, c_2, \ldots, c_n$ to C'. If not, then:

   - Add $c_{n-j}, c_{n-j+1}, \ldots, c_n$ to C', setting the rank of c_ℓ to $n - \ell$.

   - Add $c_1, c_2, \ldots, c_{n-j-1}$ to I'.

8. For each $c \in C_r$, compute $m = \Delta(c)$. If $m \geq n$, add $c_1, c_2, \ldots, c_n$ to C', setting each item to rank 0. If not, then:

   - Add $c_1, c_2, \ldots, c_{j+1}$ to C', setting each rank to 0.

109

- Add $c_{j+2}, c_{j+3}, \ldots, c_n$ to I'.

9. For each $c \in C_{dr}$, compute $j = \Gamma(c), m = \Delta(c)$. Then if $j > n - 1$, add c_1 to C', setting its rank to $n - 1$, otherwise add it to I'. Conversely, if $m > n - 1$, add c_n to C', setting its rank to 0, otherwise add it to O'. In the case where both c_1 and c_n are added to C', we have actually added one too many items to C' as there will be a duplication. This is addressed in the final step, where the in-out names of the control wires are reconciled.

10. Reconcile the C' names: For each $c_h \in C'$, compute $x = b_{cio}^{Rank(c)}(c)$. In C', replace x_n with c_h. After completing this computation, remove any duplicates.

5.2.5 Examples of folding

Example 5.2.12. Consider $S = (\_, \{a, b\}, \{c\}, \{d\}, \_, \_, \_)$ and $b_{cio} = \{(a, c), (b, a)\}$ and an iteration count of 3. See .
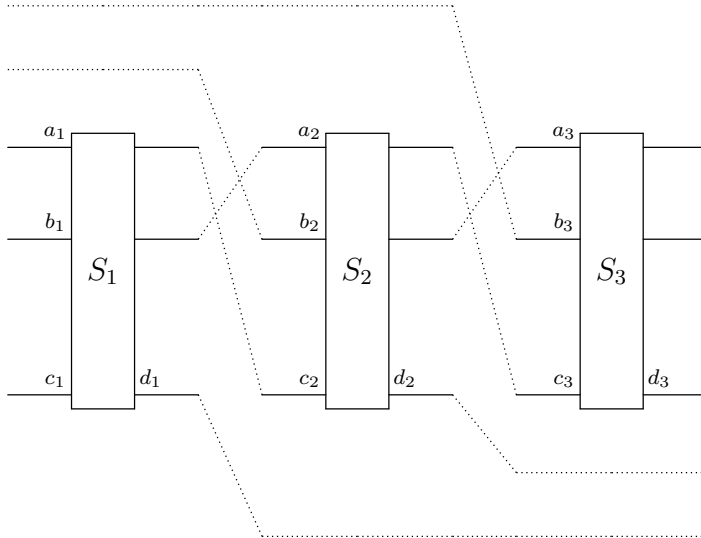
Figure 5.2: Fold with extra in/out

From the data, we compute:

$$\Omega(a) = @ \text{ and } \Omega(b) = a,$$

$$\Gamma(a) = 0 \text{ and } \Gamma(b) = 1,$$

$$\Theta(b) = * \text{ and } \Delta(b) = \infty,$$

$$\Theta(a) = b \text{ and } \Delta(a) = \infty.$$

Now, following the steps of the algorithm,

1. $I' \mapsto \{c_1\}$.

2. No change to I'.

3. $O' \mapsto \{d_3\}$.

4. $O' \mapsto \{d_1, d_2, d_3\}$.

5. $C_* = \phi, C_d = \{b\}, C_r = \phi, C_{dr} = \{a\}$.

6. For C_d, As $\Gamma(b) = 1$, we have $C' \mapsto \{b_2, b_3\}$ and $I' \mapsto \{c_1, b_1\}$. The rank of b_2 is 1 and the rank of b_3 is 0.

7. For $C_{dr} = \{a\}$, $\Gamma(a) = 0$ and $\Delta(a) = \infty$, therefore we add a_1 to I' and a_3 to C'. The rank of a_3 is zero. At this stage, we now have $I' = \{c_1, b_1, a_1\}, O' = \{d_1, d_2, d_3\}$ and $C' = \{b_2, b_3, a_3\}$.

8. Reconcile C': $b_{cio}(b) = a$ and $b_{cio}(a) = c$, therefore $b_{cio}^1(b_2) = a_3$, $b_{cio}^0(b_3) = b_3$, and $b_{cio}^0(a_3) = a_3$. Following our replacement scheme, $C' = \{b_2, b_3\}$.

Hence our final result is:

$$I' = \{c_1, b_1, a_1\},$$

$$O' = \{d_1, d_2, d_3\} \text{ and}$$

$$C' = \{b_2, b_3\}.$$

Example 5.2.13. Consider $S = (\_, \{a, b\}, \{c\}, \{d\}, \_, \_, \_)$ as in 5.2.12 on page 110 and $b_{cio} = \{(a, c), (b, a), (d, b)\}$ and an iteration count of 3. See figure 5.3 on page 112.
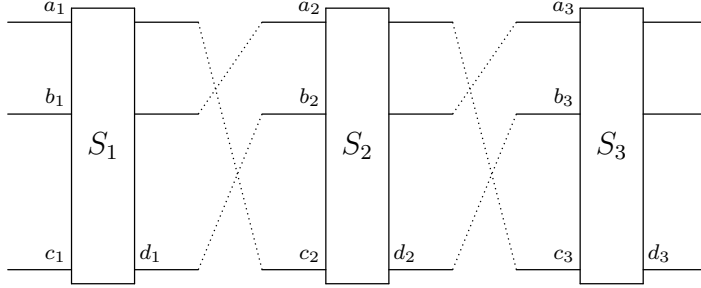
Figure 5.3: Fold with three iterations

From the data, we have:

$$\Omega(a) = @ \text{ and} \qquad\qquad \Gamma(a) = 0,$$

$$\Omega(b) = a \text{ and} \qquad\qquad \Gamma(b) = 1,$$

$$\Theta(b) = @ \text{ and} \qquad\qquad \Delta(b) = 0,$$

$$\Theta(a) = b \text{ and} \qquad\qquad \Delta(a) = 1.$$

Now, following the steps of the algorithm,

1. $I' \mapsto \{c_1\}$.

2. No change to I'.

3. $O' \mapsto \{d_3\}$.

4. All o are in the range of b_{cio}, therefore no further change to O'.

5. $C_* = C_d = C_r = \phi, C_{dr} = \{a, b\}$.

6. As $n - 1 = 2 > \Gamma(a), \Gamma(b)$, we have $I' \mapsto \{c_1, a_1, b_1\}$. Similarly, since $n - 1 = 2 > \Delta(a), \Delta(b), O' \mapsto \{a_3, b_3, d_3\}$

112

7. With C' empty, there are no further steps.

Hence our final result is:

$$I' = \{c_1, b_1, a_1\},$$

$$O' = \{a_3, b_3, d_3\} \text{ and}$$

$$C' = \phi.$$

Example 5.2.14. Consider $C = (\_, \{r, a\}, \{b, c\}, \{d, e\}, \_, \_, \_)$, $b_{cio} = \{(e, r)\}$ and an iteration count of 4. See figure 5.4 on page 113.
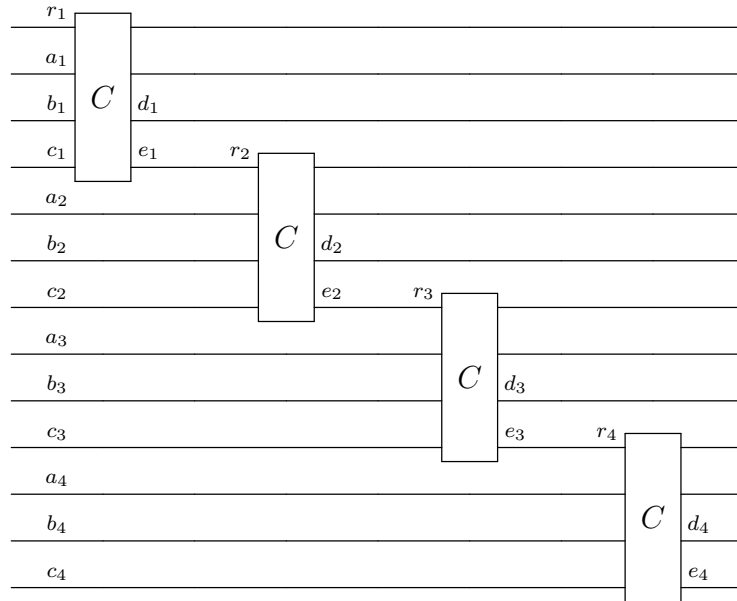
Figure 5.4: Fold of Carry

From the data, we have:

$$\Omega(r) = * \text{ and} \qquad \qquad \Omega(a) = *,$$

$$\Gamma(r) = \infty \text{ and} \qquad \qquad \Gamma(a) = \infty,$$

$$\Theta(r) = @ \text{ and} \qquad \qquad \Delta(r) = 0,$$

$$\Theta(a) = * \text{ and } \Delta(a) = \infty.$$

Now, following the steps of the algorithm,

1. $I' \mapsto \{b_1, c_1\}$.

2. No element of I is in b_{cio}, therefore we have $I' \mapsto \{b_1, c_1, b_2, c_2, b_3, c_3, b_4, c_4\}$

3. $O' \mapsto \{d_4, e_4\}$.

4. Only e is in the range of b_{cio}, therefore we add d_1, d_2, d_3 to O', giving us
   $\{d_1, d_2, d_3, d_4, e_4\}$.

5. $C_* = \{a\}, C_d = \phi, C_r = \{r\}, \ C_{dr} = \phi$.

6. Considering C_*, we add $\{a_1, a_2, a_3, a_4\}$ to C', each with rank 0.

7. Next considering C_r, as $\Delta(r) = 0$, we add r_1 to C' with rank 0 and then add
   r_2, r_3, r_4 to O'.

8. As each element of C' is of rank 0, there is no changes to the names.

Hence our final result is:

$$I' = \{b_1, c_1, b_2, c_2, b_3, c_3, b_4, c_4\},$$

$$O' = \{d_1, d_2, d_3, d_4, e_4, r_2, r_3, r_4\} \text{ and}$$

$$C' = \{r_1, a_1, a_2, a_3, a_4\}.$$

5.3 Alternate Algorithm for Fold Transformation

In this section, we present an alternate algorithm for calculating the type and names of control, input and output wires for a folded subroutine. The starting point is 5.2.11 on page 107 and the ordered pairs of b_{cio}: A set of ordered pairs $\{(f,t)|f \in C \cup I, t \in C \cup O\}$.

To implement the algorithm of calculating the folded subroutines C, I and O, we need the following:

- $I \cap O = \phi$, which may be accomplished by renaming if needed.

- We create the sets C_j, I_j, O_j for $j = 1 \ldots n$ where the members of X_j are the elements of X together with the subscript j where X is one of C, I, O.

The algorithm proceeds by creating a set of "type" constraints for each of the elements of the new sets, based upon b_{cio} and membership in a C, I or O set.

The algorithm steps are:

1. Create a set $\mathcal{C}$ of pairs (a_j, b_{j+1}) for j ranging from 0 to $n-1$ based upon the bijection b_{cio}.

2. For each $j = 0 \ldots n - 1$,

   (a) For all $c \in C_j$, if $(c, \_) \notin \mathcal{C}$, add the pair (c, ϕ)

   (b) For all $c \in C_j$, if $(\_, c) \notin \mathcal{C}$, add the pair (ϕ, c)

   (c) For all $o \in O_j$, if $(o, \_) \notin \mathcal{C}$, add the pair (o, ϕ)

   (d) For all $i \in I_j$, if $(\_, i) \notin \mathcal{C}$, add the pair (ϕ, i)

3. Convert the pairs in $\mathcal{C}$ to equations and constraints as follows for each pair (a, b):

   (a) $a = \phi \implies \begin{cases} b :: (b, C) & \text{when } b \in C_j \text{for some } j, \\ b :: (b, I) & \text{when } b \in I_j \text{for some } j. \end{cases}$

   (b) $a \in C_{j-1} \implies \begin{cases} b = a & \text{when } b \in C_j, \\ b = a, b :: I & \text{when } b \in I_j, \\ a :: (C, a) & \text{when } b = \phi. \end{cases}$

$$
\text{(c) } a \in O_{j-1} \implies
\begin{cases}
b = a, b :: O & \text{when } b \in C_j, \\[2mm]
\text{no equation} & \text{when } b \in I_j, \\[2mm]
a :: (O, a) & \text{when } b = \phi.
\end{cases}
$$

4. Solve the equations with the constraints, with the assumption that
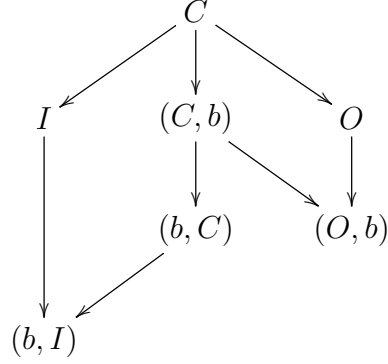
For example,

- $a :: O, a :: C$ is solvable with $a :: O$;

- $a = b, b :: (b, C), a :: I$ is solved by $b, a :: (b, I)$;

- $a = b, b = d, a :: (a, C), d :: (C, d)$ is solved by $a, b, d :: (a, C)$.

5. For the folded subroutine, X will be all items of "type" X, where X is any of C, I, O, the names of each entry will be the companion name with the "type".

5.3.1 Examples of folding with Alternate Algorithm

Example 5.3.1. We repeat 5.2.12 on page 110, that is, $S = (\_, \{a, b\}, \{c\}, \{d\}, \_, \_, \_)$ and $b_{cio} = \{(a, c), (b, a), (d, b)\}$ and an iteration count of 3. See figure 5.3 on the preceding page.

There are two control wires a, b, an input wire d and one output wire d. We will do three

iterations. Our set $\mathcal{C}$ of pairs becomes:

$$\{(\phi, a), \ (\phi, b), \ (\phi, c),$$
$$(b, a_1), \ (d, b_1), \ (a, c_1),$$
$$(b_1, a_2), \ (d_1, b_2), \ (a_1, c_2),$$
$$(a_2, \phi), \ (b_2, \phi), \ (c_2, \phi)\}.$$

Translating each of these to equations and constraints, we get:

$$a :: (a, C), b :: (b, C), c :: (c, I)$$

$$b = a_1, b_1 = d \ b_1 :: O, a = c_1 \ c_1 :: I$$

$$b_1 = a_2, b_2 = d_1 \ b_2 :: O, a_1 = c_2 \ c_2 :: I$$

$$a_2 :: (C, a_2), b_2 :: (C, b_2), d_2 :: (O, d_2)$$

As we have three wires in and three wires out, we expect to see six separate groupings in the solution to the above equations. We will proceed by starting from the top left.

start: a	$a = c_1, :: I, :: (a, C)$	$\implies a :: (a, I)$
start: b	$b = a_1, a_1 = c_2, :: I, :: (b, C)$	$\implies b :: (b, I)$
start: c	$:: (c, I)$	$\implies c :: (c, I)$
start: b_1	$b_1 = d, b_1 = a_2, :: O, :: (C, a_2)$	$\implies a_2 :: (O, a_2)$
start: b_2	$b_2 = d_1, :: O, :: (C, b_2)$	$\implies b_2 :: (O, b_2)$
start: d_2	$:: (O, d_2)$	$\implies d_2 :: (O, d_2)$

Chapter 6

Synthesis of quantum operations

6.1 Introduction to synthesis

An important problem in quantum information theory is the decomposition of arbitrary unitary operators into gates from some fixed universal set [17]. Depending on the operator to be decomposed, this may either be done exactly or to within some given accuracy ϵ; the former problem is known as *exact synthesis* and the latter as *approximate synthesis* [12].

6.2 Algebraic background

We first introduce some notation and terminology, following [12] where possible. Recall that $\mathbb{N}$ is the set of natural numbers including 0, and $\mathbb{Z}$ is the ring of integers. We write $\mathbb{Z}_2 = \mathbb{Z}/2\mathbb{Z}$ for the ring of integers modulo 2. Let $\mathbb{D}$ be the ring of *dyadic fractions*, defined as $\mathbb{D} = \mathbb{Z}[\frac{1}{2}] = \{\frac{a}{2^n} \mid a \in \mathbb{Z}, n \in \mathbb{N}\}$.

Let $\omega = e^{i\pi/4} = (1+i)/\sqrt{2}$. Note that ω is an 8th root of unity satisfying $\omega^2 = i$ and $\omega^4 = -1$. We will consider three different rings related to ω:

Definition 6.2.1. Consider the following rings. Note that the first two are subrings of the complex numbers, and the third one is not:

- $\mathbb{D}[\omega] = \{a\omega^3 + b\omega^2 + c\omega + d \mid a, b, c, d \in \mathbb{D}\}$.

- $\mathbb{Z}[\omega] = \{a\omega^3 + b\omega^2 + c\omega + d \mid a, b, c, d \in \mathbb{Z}\}$.

- $\mathbb{Z}_2[\omega] = \{p\omega^3 + q\omega^2 + r\omega + s \mid p, q, r, s \in \mathbb{Z}_2\}$.

Note that the ring $\mathbb{Z}_2[\omega]$ only has 16 elements. The laws of addition and multiplication are uniquely determined by the ring axioms and the property $\omega^4 = 1 \pmod{2}$. We call the

elements of $\mathbb{Z}_2[\omega]$ *residues* (more precisely, residue classes of $\mathbb{Z}[\omega]$ modulo 2).

Remark 6.2.2. The ring $\mathbb{D}[\omega]$ is the same as the ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$. However, as already pointed out in [12], the formulation in terms of ω is far more convenient algebraically.

Remark 6.2.3. The ring $\mathbb{Z}[\omega]$ is also called the *ring of algebraic integers* of $\mathbb{D}[\omega]$. It has an intrinsic definition, i.e., one that is independent of the particular presentation of $\mathbb{D}[\omega]$. Namely, a complex number is called an *algebraic integer* if it is the root of some polynomial with integer coefficients and leading coefficient 1. It follows that ω, i, and $\sqrt{2}$ are algebraic integers, whereas, for example, $1/\sqrt{2}$ is not. The ring $\mathbb{Z}[\omega]$ then consists of precisely those elements of $\mathbb{D}[\omega]$ that are algebraic integers.

6.2.1   Conjugate and norm

Remark 6.2.4 (Complex conjugate and norm)**.** Since $\mathbb{D}[\omega]$ and $\mathbb{Z}[\omega]$ are subrings of the complex numbers, they inherit the usual notion of complex conjugation. We note that $\omega^\dagger = -\omega^3$. This yields the following formula:

$$(a\omega^3 + b\omega^2 + c\omega + d)^\dagger = -c\omega^3 - b\omega^2 - a\omega + d. \tag{6.1}$$

Similarly, the sets $\mathbb{D}[\omega]$ and $\mathbb{Z}[\omega]$ inherit the usual norm from the complex numbers. It is given by the following explicit formula, for $t = a\omega^3 + b\omega^2 + c\omega + d$:

$$\|t\|^2 = t^\dagger t = (a^2 + b^2 + c^2 + d^2) + (cd + bc + ab - da)\sqrt{2}. \tag{6.2}$$

Definition 6.2.5 (Weight)**.** For $t \in \mathbb{D}[\omega]$ or $t \in \mathbb{Z}[\omega]$, the *weight* of t is denoted $\|t\|_{\text{weight}}$, and is given by:

$$\|t\|^2_{\text{weight}} = a^2 + b^2 + c^2 + d^2. \tag{6.3}$$

Note that the square of the norm is valued in $\mathbb{D}[\sqrt{2}]$, whereas the square of the weight is valued in $\mathbb{D}$. We also extend the definition of norm and weight to vectors in the obvious

way: For $u = (u_j)_j$, we define

$$\|u\|^2 = \sum_j \|u_j\|^2 \quad \text{and} \quad \|u\|^2_{\text{weight}} = \sum_j \|u_j\|^2_{\text{weight}}.$$

Lemma 6.2.6. *Consider a vector $u \in \mathbb{D}[\omega]^n$. If $\|u\|^2$ is an integer, then $\|u\|^2_{\text{weight}} = \|u\|^2$.*

Proof. Any $t \in \mathbb{D}[\sqrt{2}]$ can be uniquely written as $t = a + b\sqrt{2}$, where $a, b \in \mathbb{D}$. We can call a the *dyadic part* of t. Now the claim is obvious, because $\|u\|^2_{\text{weight}}$ is exactly the dyadic part of $\|u\|^2$. $\qquad\square$

6.2.2 Denominator exponents

Definition 6.2.7. Let $t \in \mathbb{D}[\omega]$. A natural number $k \in \mathbb{N}$ is called a *denominator exponent* for t if $\sqrt{2}^k t \in \mathbb{Z}[\omega]$. It is obvious that such k always exists. The least such k is called the *least denominator exponent* of t.

More generally, we say that k is a denominator exponent for a vector or matrix if it is a denominator exponent for all of its entries. The least denominator exponent for a vector or matrix is therefore the least k that is a denominator exponent for all of its entries.

Remark 6.2.8. Our notion of least denominator exponent is almost the same as the "smallest denominator exponent" of [12], except that we do not permit $k < 0$.

6.2.3 Residues

Remark 6.2.9. The ring $\mathbb{Z}_2[\omega]$ is not a subring of the complex numbers; rather, it is a quotient of the ring $\mathbb{Z}[\omega]$. Indeed, consider the *parity function* $\overline{()} : \mathbb{Z} \to \mathbb{Z}_2$, which is the unique ring homomorphism. It satisfies $\bar{a} = 0$ if a is even and $\bar{a} = 1$ if a is odd. The parity map induces a surjective ring homomorphism $\rho : \mathbb{Z}[\omega] \to \mathbb{Z}_2[\omega]$, defined by

$$\rho(a\omega^3 + b\omega^2 + c\omega + d) = \bar{a}\omega^3 + \bar{b}\omega^2 + \bar{c}\omega + \bar{d}.$$

We call ρ the *residue map*, and we call $\rho(t)$ the *residue* of t.

$\rho(t)$	$\rho(\sqrt{2}\,t)$	$\rho(t^\dagger t)$	$\rho(t)$	$\rho(\sqrt{2}\,t)$	$\rho(t^\dagger t)$
0000	0000	0000	1000	0101	0001
0001	1010	0001	1001	1111	1010
0010	0101	0001	1010	0000	0000
0011	1111	1010	1011	1010	0001
0100	1010	0001	1100	1111	1010
0101	0000	0000	1101	0101	0001
0110	1111	1010	1110	1010	0001
0111	0101	0001	1111	0000	0000

Table 6.1: Some operations on residues

Convention 6.2.10. Since residues will be important for the constructions of this thesis, we introduce a shortcut notation, writing each residue $p\omega^3 + q\omega^2 + r\omega + s$ as a string of binary digits $pqrs$.

What makes residues useful for our purposes is that many important operations on $\mathbb{Z}[\omega]$ are well-defined on residues. Here, we say that an operation $f : \mathbb{Z}[\omega] \to \mathbb{Z}[\omega]$ is *well-defined on residues* if for all t, s, $\rho(t) = \rho(s)$ implies $\rho(f(t)) = \rho(f(s))$.

For example, two operations that are obviously well-defined on residues are complex conjugation, which takes the form $(pqrs)^\dagger = rqps$ by (equation (6.1) on page 119), and multiplication by ω, which is just a cyclic shift $\omega(pqrs) = qrsp$. Table table 6.1 on page 121 shows two other important operations on residues, namely multiplication by $\sqrt{2}$ and the squared norm.

Definition 6.2.11 (*k*-Residue). Let $t \in \mathbb{D}[\omega]$ and let k be a (not necessarily least) denominator exponent for t. The *k-residue of t*, in symbols $\rho_k(t)$, is defined to be

$$\rho_k(t) = \rho(\sqrt{2}^k t).$$

Definition 6.2.12 (Reducibility). We say that a residue $x \in \mathbb{Z}_2[\omega]$ is *reducible* if it is of the form $\sqrt{2}\,y$, for some $y \in \mathbb{Z}_2[\omega]$. Moreover, we say that $x \in \mathbb{Z}_2[\omega]$ is *twice reducible* if it is of the form $2y$, for some $y \in \mathbb{Z}_2[\omega]$.

Lemma 6.2.13. *For a residue x, the following are equivalent:*

(a) x is reducible;

(b) $x \in \{0000, 0101, 1010, 1111\}$;

(c) $\sqrt{2}\,x = 0000$;

(d) $x^\dagger x = 0000$.

Moreover, x is twice reducible iff $x = 0000$.

Proof. By inspection of Table . $\qquad\square$

Lemma 6.2.14. *Let $t \in \mathbb{Z}[\omega]$. Then $t/2 \in \mathbb{Z}[\omega]$ if and only if $\rho(t)$ is twice reducible, and $t/\sqrt{2} \in \mathbb{Z}[\omega]$ if and only if $\rho(t)$ is reducible.*

Proof. The first claim is trivial, as $\rho(t) = 0000$ if and only if all components of t are even. For the second claim, the left-to-right implication is also trivial: assume $t' = t/\sqrt{2} \in \mathbb{Z}[\omega]$. Then $\rho(t) = \rho(\sqrt{2}\,t')$, which is reducible by definition. Conversely, let $t \in \mathbb{Z}[\omega]$ and assume that $\rho(t)$ is reducible. Then $\rho(t) \in \{0000, 0101, 1010, 1111\}$, and it can be seen from Table that $\rho(\sqrt{2}\,t) = 0000$. Therefore, $\sqrt{2}\,t$ is twice reducible by the first claim; hence t is reducible.

$\qquad\square$

Corollary 6.2.15. *Let $t \in \mathbb{D}[\omega]$ and let $k > 0$ be a denominator exponent for t. Then k is the least denominator exponent for t if and only if $\rho_k(t)$ is irreducible.*

Proof. Since k is a denominator exponent for t, we have $\sqrt{2}^k t \in \mathbb{Z}[\omega]$. Moreover, k is least if and only if $\sqrt{2}^{k-1} t \notin \mathbb{Z}[\omega]$. By Lemma , this is the case if and only if $\rho(\sqrt{2}^k t) = \rho_k(t)$ is irreducible. $\qquad\square$

Lemma 6.2.16. *For all a in $\mathbb{Z}[\omega]$, $a + a^t$ is divisible by $\sqrt{2}$ in $\mathbb{Z}[\omega]$.*

Proof. It is sufficient to show this on the generators $\{1, \omega, \omega^2, \omega^3\}$.

1. $1 + 1 = 2$ and $\frac{2}{\sqrt{2}} = \sqrt{2} = \omega - \omega^3$.

2. $\omega + \omega^t = \omega - \omega^3 = \sqrt{2}$.

3. $\omega^2 + (\omega^2)^t = \omega^2 - \omega^2 = 0$.

4. $\omega^3 + (\omega^3)^t = \omega^3 - \omega = -\sqrt{2}$.

$\square$

Definition 6.2.17. The notions of residue, k-residue, reducibility, and twice-reducibility all extend in an obvious componentwise way to vectors and matrices. Thus, the residue $\rho(u)$ of a vector or matrix u is obtained by taking the residue of each of its entries, and similar for k-residues. Also, we say that a vector or matrix is reducible if each of its entries is reducible, and similarly for twice-reducibility.

Example 6.2.18. Consider the matrix

$$U = \frac{1}{\sqrt{2}^3} \begin{pmatrix} -\omega^3 + \omega - 1 & \omega^2 + \omega + 1 & \omega^2 & -\omega \\ \omega^2 + \omega & -\omega^3 + \omega^2 & -\omega^2 - 1 & \omega^3 + \omega \\ \omega^3 + \omega^2 & -\omega^3 - 1 & 2\omega^2 & 0 \\ -1 & \omega & 1 & -\omega^3 + 2\omega \end{pmatrix}.$$

It has least denominator exponent 3. Its 3-, 4-, and 5-residues are:

$$\rho_3(U) = \begin{pmatrix} 1011 & 0111 & 0100 & 0010 \\ 0110 & 1100 & 0101 & 1010 \\ 1100 & 1001 & 0000 & 0000 \\ 0001 & 0010 & 0001 & 1000 \end{pmatrix},$$

$$\rho_4(U) = \begin{pmatrix} 1010 & 0101 & 1010 & 0101 \\ 1111 & 1111 & 0000 & 0000 \\ 1111 & 1111 & 0000 & 0000 \\ 1010 & 0101 & 1010 & 0101 \end{pmatrix}, \quad \rho_5(U) = 0.$$

6.3 Exact synthesis of single qubit operators

Matsumoto and Amano [15] showed that every single-qubit Clifford+T operator can be uniquely written in the following form, which we call the *Matsumoto-Amano normal form*:

$$(T \mid \varepsilon)\,(HT \mid SHT)^*\,\mathcal{C}. \tag{6.4}$$

Here, we have used the syntax of regular expressions [11] to denote a set of sequences of operators. The symbol ε denotes the empty sequence (more precisely, the singleton set containing just the empty sequence); if $\mathcal{L}$ and $\mathcal{K}$ are two sets of sequences, then $\mathcal{L} \mid \mathcal{K}$ denotes their union; $\mathcal{L}\mathcal{K}$ denotes the set $\{st \mid s \in \mathcal{L}, t \in \mathcal{K}\}$; $\mathcal{L}^*$ denotes the set $\{s_1 \dots s_n \mid n \geq 0; s_1, \dots, s_n \in \mathcal{L}\}$; and $\mathcal{C}$ denotes any Clifford operator. In words, the Matsumoto-Amano representation of an operator consists of a Clifford operator, followed by any number of *syllables* of the form HT or SHT, followed by an optional syllable T. (We follow the usual convention of multiplying operators right-to-left, so when we say one operator "follows" another, we mean that it appears to its left).

The most important properties of the Matsumoto-Amano decomposition are:

- Existence: every single-qubit Clifford+T operator can be written in Matsumoto-Amano normal form (moreover, there is an efficient algorithm for converting the operator to normal form);

- Uniqueness: no operator can be written in Matsumoto-Amano normal form in more than one way;

- T-optimality: of all the possible exact decompositions of a given operator into the Clifford+T set of gates, the Matsumoto-Amano normal form contains the smallest possible number of T-gates.

It is perhaps less well-known that the uniqueness proof given by Matsumoto and Amano yields an efficient *algorithm* for T-optimal exact single-qubit synthesis. One may contrast this, for example, with the recent algorithm by Kliuchnikov et al. [12], which is efficient, but only asymptotically T-optimal. The purpose of this note is to give a detailed presentation of the algorithmic content of Matsumoto and Amano's result. Along the way, we also simplify Matsumoto and Amano's proofs, and we give an intrinsic characterization of the Clifford+T subgroup of $SO(3)$.

6.3.1 Existence

In the following, we will often speak of sequences of operators. For our purposes, a sequence is just an n-tuple. We write st for the operation of concatenating two sequences, and we write ε for the empty sequence. We identify a 1-tuple (A) with the operator A itself. If $s = (A_1, \ldots, A_n)$ is a sequence of operators, we write $[\![s]\!] = A_1 \cdots A_n$ for the product of the operators in the sequence; naturally, $[\![\varepsilon]\!] = I$. Note that the notation is ambiguous; for example, depending on the context, SHT may denote either the sequence (S, H, T) of 3 operators, or their product, which is a single operator. To alleviate the ambiguity, we assume that everything is a sequence by default, and we write $s \equiv t$ if two sequences are equal as tuples, and $s = t$ if they are equal as operators, i.e., if $[\![s]\!] = [\![t]\!]$.

Remark 6.3.1. Consider any operator A in Matsumoto-Amano normal form. If λ is any unit scalar, then λA can clearly also be written in Matsumoto-Amano normal form with the same T-count, namely by multiplying λ into the rightmost Clifford operator. Moreover, if A can be *uniquely* written in Matsumoto-Amano normal form, then the same is true for λA. Therefore, nothing is added or lost to the Matsumoto-Amano normal form whether one allows arbitrary global phases, a suitable discrete set of global phases (for example, powers of $e^{i\pi/4}$), or whether one works modulo global phase. Since it is convenient to work modulo global phase, we do so in the remainder; however, this does not restrict the generality of the results.

Definition 6.3.2. Let $\mathcal{C}$ denote the Clifford group on one qubit, modulo global phases. This group has 24 elements. Let $\mathcal{S}$ be the 8-element subgroup spanned by S and X. Let $\mathcal{C}' = \mathcal{C} \setminus \mathcal{S}$. Let $\mathcal{H} = \{I, H, SH\}$ and $\mathcal{H}' = \{H, SH\}$.

Lemma 6.3.3. *The following hold:*

$$\mathcal{C} = \mathcal{HS}, \tag{6.5}$$

$$\mathcal{C}' = \mathcal{H}'\mathcal{S}, \tag{6.6}$$

$$\mathcal{SH}' \subseteq \mathcal{H}'\mathcal{S}, \tag{6.7}$$

$$\mathcal{S}T = T\mathcal{S}, \tag{6.8}$$

$$T\mathcal{S}T = \mathcal{S}. \tag{6.9}$$

Proof. Since $\mathcal{S}$ is an 8-element subgroup of $\mathcal{C}$, it has three left cosets. They are $\mathcal{S}$, $H\mathcal{S}$, and $SH\mathcal{S}$. Since $\mathcal{C}$ is the disjoint union of these cosets, (equation (6.5) on page 126) and (equation (6.6) on page 126) immediately follow. For (equation (6.7) on page 126), first notice that $\mathcal{S}S = \mathcal{S}$, and therefore $\mathcal{SH}' = \mathcal{S}H \cup \mathcal{S}SH = \mathcal{S}H$. Since $\mathcal{S}H$ is a non-trivial right coset of $\mathcal{S}$, it follows that $\mathcal{S}H \subseteq \mathcal{C} \setminus \mathcal{S} = \mathcal{C}'$. Combining these facts with (equation (6.6)), we have (equation (6.7)). Finally, the equations (equation (6.8)) and (equation (6.9)) are trivial consequences of the equations $ST = TS$, $XT = TXS$, and $TT = S$. $\qquad\square$

Proposition 6.3.4 (Matsumoto and Amano [15])**.** *Every single-qubit Clifford+T operator can be written in Matsumoto-Amano normal form.*

Proof. Let M be a single-qubit Clifford+T operator. Clearly, M can be written as

$$M = C_n \, T \, C_{n-1} \, \cdots \, C_1 \, T \, C_0, \tag{6.10}$$

for some $n \geq 0$, where $C_0, \ldots, C_n \in \mathcal{C}$. First note that if $C_i \in \mathcal{S}$ for any $i \in \{1, \ldots, n-1\}$, then we can immediately use (equation (6.9)) to replace TC_iT by a single Clifford operator. This yields a shorter expression of the form (equation (6.10)) for M. We may therefore assume without loss of generality that $C_i \notin \mathcal{S}$ for $i = 1, \ldots, n-1$. If $n = 0$, then M is a

Clifford operator, and there is nothing to show. Otherwise, we have

$$M \in \mathcal{C} \, T \, \mathcal{C}' \, \cdots \, \mathcal{C}' \, T \, \mathcal{C} \qquad \text{by (equation (6.10))} \tag{6.11}$$

$$= \mathcal{H}\mathcal{S} \, T \, \mathcal{H}'\mathcal{S} \, \cdots \, \mathcal{H}'\mathcal{S} \, T \, \mathcal{C} \qquad \text{by (equation (6.5)) and (equation (6.6))} \tag{6.12}$$

$$\subseteq \mathcal{H} \, T \, \mathcal{H}' \, \cdots \, \mathcal{H}' \, T \, \mathcal{C} \qquad \text{by (equation (6.7)) and (equation (6.8)).} \tag{6.13}$$

Note how, in the last step, the relations (equation (6.7)) and (equation (6.8)) were used to move all occurrences of $\mathcal{S}$ to the right, where they were absorbed into the final $\mathcal{C}$. It is now trivial to see that every element of (equation (6.13) on the next page) can be written in Matsumoto-Amano normal form, finishing the proof. □

Corollary 6.3.5. *There exists a linear-time algorithm for symbolically reducing any sequence of Clifford+T operators to Matsumoto-Amano normal form. More precisely, this algorithm runs in time at most $O(n)$, where n is the length of the input sequence.*

Proof. The proof of Proposition 6.3.4 on the preceding page already contains an algorithm for reducing any sequence of Clifford+T operators to Matsumoto-Amano normal form. However, in the stated form, it is perhaps not obvious that the algorithm runs in linear time. Indeed, a naive implementation of the first step would require up to n searches of the entire sequence for a term of the form $T\mathcal{S}T$, which can take time $O(n^2)$.

One obtains a linear time algorithm from the following observation: if M is already in Matsumoto-Amano normal form, and A is either a Clifford operator or T, then MA can be reduced to Matsumoto-Amano normal form in constant time. This is trivial when A is a Clifford operator, because it will simply be absorbed into the rightmost Clifford operator of M. In the case where $A = T$, a simple case distinction shows that at most the rightmost 5 elements of MA need to be updated. The normal form of a sequence of operators $A_1 A_2 \ldots A_n$ can now be computed by starting with $M = I$ and repeatedly right-multiplying by $A_1, \ldots, A_n$, reducing to normal form after each step. □

6.3.2 *T*-Optimality

Corollary 6.3.6. *Let M be an operator in the Clifford+T group, and assume that M can be written with T-count n. Then there exists a Matsumoto-Amano normal form for M with T-count at most n.*

Proof. This is an immediately consequence of the proof of Proposition 6.3.4 on the previous page, because the reduction from (equation (6.10) on the preceding page) to (equation (6.13)) does not increase the *T*-count. □

6.3.3 Uniqueness

Theorem 6.3.7 (Matsumoto and Amano [15]). *If M and N are two different Matsumoto-Amano normal forms, then they describe different operators.*

Recall that each single-qubit unitary operator (modulo global phase) can be uniquely represented as a rotation on the Bloch sphere, or equivalently, as an element of $SO(3)$, the real orthogonal 3×3 matrices with determinant 1. The relationship between an operator $U \in U(2)$ and its Bloch sphere representation $\hat{U} \in SO(3)$ is given by

$$\hat{U}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} \iff U(xX + yY + zZ)U^\dagger = x'X + y'Y + z'Z, \qquad (6.14)$$

where X, Y, and Z are the Pauli operators. The Bloch sphere representations of the operators H, S, and T are:

$$\hat{H} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \quad \hat{S} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \hat{T} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & \sqrt{2} \end{pmatrix}. \qquad (6.15)$$

Remark 6.3.8. The Bloch sphere representation of any scalar is the identity matrix. The Bloch sphere representations of the 24 Clifford operators (modulo phase) are precisely those elements of $SO(3)$ that can be written with matrix entries in $\{-1, 0, 1\}$; these are exactly the 24 symmetries of the cube $\{(x, y, z) \mid -1 \le x, y, z \le 1\}$.

Definition 6.3.9. Recall that $\mathbb{N}$ denotes the natural numbers including 0; $\mathbb{Z}$ denotes the integers; and $\mathbb{Z}_2$ denotes the integers modulo 2. We define three subrings of the real numbers:

- $\mathbb{D} = \mathbb{Z}[\frac{1}{2}] = \{\frac{a}{2^n} \mid a \in \mathbb{Z}, n \in \mathbb{N}\}$. This is the ring of *dyadic fractions*.

- $\mathbb{Z}[\sqrt{2}] = \{a + b\sqrt{2} \mid a, b \in \mathbb{Z}\}$. This is the ring of *quadratic integers* with radicand 2.

- $\mathbb{D}[\sqrt{2}] = \mathbb{Z}[\frac{1}{\sqrt{2}}] = \{r + s\sqrt{2} \mid r, s \in \mathbb{D}\}$.

We will also need a fourth ring, which is not a subring of the real numbers.

- $\mathbb{Z}_2[\sqrt{2}] = \{a + b\sqrt{2} \mid a, b \in \mathbb{Z}_2\}$.

Note that the ring $\mathbb{Z}_2[\sqrt{2}]$ has only 4 elements; they are residue classes modulo 2 of the ring $\mathbb{Z}[\sqrt{2}]$. For brevity, we refer to the elements of $\mathbb{Z}_2[\sqrt{2}]$ as *residues*.

Definition 6.3.10 (Residue[1] and parity). Consider the unique ring homomorphism $\mathbb{Z} \to \mathbb{Z}_2$, mapping $a \in \mathbb{Z}$ to $\bar{a} \in \mathbb{Z}_2$, where $\bar{a} = 0$ if a is even and $\bar{a} = 1$ if a is odd. This induces a surjective ring homomorphism $\rho : \mathbb{Z}[\sqrt{2}] \to \mathbb{Z}_2[\sqrt{2}]$, defined by $\rho(a + b\sqrt{2}) = \bar{a} + \bar{b}\sqrt{2}$. For any given $x \in \mathbb{Z}[\sqrt{2}]$, we refer to $\rho(x)$ as the *residue of x*.

Moreover, consider the ring homomorphism $p : \mathbb{Z}[\sqrt{2}] \to \mathbb{Z}_2$ given by $p(a + b\sqrt{2}) = \bar{a}$. We refer to $p(x)$ as the *parity* of x.

Definition 6.3.11 (Denominator exponent). For every element $q \in \mathbb{D}[\sqrt{2}]$, there exists some natural number $k \geq 0$ such that $\sqrt{2}^k q \in \mathbb{Z}[\sqrt{2}]$, or equivalently, such that q can be written as $\frac{x}{\sqrt{2}^k}$, for some quadratic integer x. Such k is called a *denominator exponent* for q. The least such k is called the *least denominator exponent* of q.

More generally, we say that k is a denominator exponent for a vector or matrix if it is a denominator exponent for all of its entries. The least denominator exponent for a vector or matrix is therefore the least k that is a denominator exponent for all of its entries.

---

[1]I don't think we need residue here, which is good as it will then be used only in section 7, the $U(2)$ case

$$\text{Start:}\quad \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \supset \mathcal{C}$$

$$k{+}{+}\,\Big\downarrow\, T$$

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \xleftarrow[k{+}{+}]{T}$$

$$H\Big\downarrow \quad \Big\uparrow\, T_{k{+}{+}}$$

$$\begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix} \xrightarrow{S} \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix}$$

Figure 6.1: The action of Matsumoto-Amano normal forms on k-parities. All matrices are written modulo the right action of the Clifford group, i.e., modulo a permutation of the columns.

Definition 6.3.12 (k-parity). Let k be a denominator exponent for $q \in \mathbb{D}[\sqrt{2}]$. We define the *$k$-residue* of q, in symbols $\rho_k(q) \in \mathbb{Z}_2[\sqrt{2}]$, and the *$k$-parity* of q, in symbols $p_k(q) \in \mathbb{Z}_2$, by

$$\rho_k(q) = \rho(\sqrt{2}^k q), \qquad p_k(q) = p(\sqrt{2}^k q).$$

The k-residue and k-parity of a vector or matrix are defined componentwise.

Remark 6.3.13. Let C be any Clifford operator, and $\hat{C}$ its Bloch sphere representation. As noted above, the matrix entries of $\hat{C}$ are in $\{-1, 0, 1\}$; it follows that $\hat{C}$ has denominator exponent 0. In particular, it follows that multiplication by $\hat{C}$ is a well-defined operation on parity matrices: for any 3×3-matrix U with entries in $\mathbb{Z}_2[\sqrt{2}]$, we define $U \bullet C := U \cdot p(\hat{C})$. This defines a right action of the Clifford group on the set of parity matrices.

Definition 6.3.14. If G is any subgroup of the Clifford group, we define $\sim_G$ the be the equivalence relation induced by this right action, i.e., for parity matrices U, V, we write $U \sim_G V$ if there exists some $C \in G$ such that $V = U \bullet G$. In case $G = \mathcal{C}$ is the entire Clifford group, $U \sim_{\mathcal{C}} V$ holds if and only if U and V differ by a permutation of columns.

Lemma 6.3.15. *Let M be a Matsumoto-Amano normal form, and $\hat{M} \in SO(3)$ the Bloch sphere operator of M. Let k be the least denominator exponent of $\hat{M}$. Then exactly one of the following holds:*

- $k = 0$, and M is a Clifford operator.

- $k > 0$, $p_k(\hat{M}) \sim_{\mathcal{C}} \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$, and the leftmost syllable in M is T.

- $k > 0$, $p_k(\hat{M}) \sim_{\mathcal{C}} \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix}$, and the leftmost syllable in M is HT.

- $k > 0$, $p_k(\hat{M}) \sim_{\mathcal{C}} \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix}$, and the leftmost syllable in M is SHT.

Moreover, the T-count of M is equal to k.

Proof. By induction on the length of the Matsumoto-Amano normal form M. Figure figure 6.1 on page 130 shows the action of Matsumoto-Amano operators on parity matrices. Each vertex represents a $\sim_{\mathcal{C}}$-equivalence class of k-parities. The vertex labelled "Start" represents the empty Matsumoto-Amano normal form, i.e., the identity operator. Each arrow represents left multiplication by the relevant operator, i.e., a Clifford operator, T, H, or S. Thus, each Matsumoto-Amano normal form, read from right to left, gives rise to a unique path in the graph of Figure figure 6.1 on page 130. The label $k++$ on an arrow indicates that the least denominator exponent increases by 1. The claims of the lemma then immediately follow from Figure figure 6.1 on page 130. □

Proof of Theorem 6.3.7 on page 128. This is an immediate consequence of Lemma 6.3.15. Indeed, suppose that M and N are two Matsumoto-Amano normal forms describing the same Bloch sphere operator U. We show that $M = N$ by induction on the length of M. Let k be the least denominator exponent of U. If $k = 0$, then by Lemma 6.3.15, both M and N

are Clifford operators; since they describe the same Bloch sphere operator, they differ only by a phase.[2] If $k > 0$, then by Lemma 6.3.15, the Matsumoto-Amano normal forms M and N have the same leftmost operator (either T, H, or S), and the claim follows by induction hypothesis. □

6.3.4 The Matsumoto-Amano decomposition algorithm

As an immediate consequence of Lemma 6.3.15 on the preceding page, we can an efficient algorithm for calculating the Matsumoto-Amano normal form of any Clifford+T operator, given as a matrix.

Theorem 6.3.16. *Let $U \in SO(3)$ be the Bloch sphere representation of some Clifford+T operator. Let k be the least denominator exponent of U. Then the Matsumoto-Amano normal form M of U can be efficiently computed with $O(k)$ arithmetic operations.*

Proof. By assumption, U is the Bloch sphere representation of some Clifford+T operator. Let M be the unique Matsumoto-Amano normal form of this operator[3]. Note that, by Lemma 6.3.15 on the previous page, the T-count of M is k. We compute M recursively. If $k = 0$, then by Lemma 6.3.15 on the preceding page, M is a Clifford operator; it can be determined from the matrix U in constant time. If $k > 0$, we compute $p_k(U)$, which must be one of the three cases listed in Lemma 6.3.15 on page 130. This determines whether the leftmost syllable of M is T, HT, or SHT. Let N be this syllable, so that $M = NM'$, for some Matsumoto-Amano normal form M'. Then M' can be recursively computed as the Matsumoto-Amano normal form of $U' = \hat{N}^{-1}U$; moreover, since M' has T-count $k - 1$, the recursion terminates after k steps. Since each induction step only requires a constant number of arithmetic operations, the total number of operations is $O(k)$. □

---

[2]Todo: fix the treatment of phases
[3]Todo: treat phase correctly

6.3.5   A characterization of Clifford+T on the Bloch sphere

Lemma 6.3.17. *Let $U \in SO(3)$ be an orthogonal matrix with entries in $\mathbb{D}[\sqrt{2}]$. Let k be a denominator exponent of U, and let v_1, v_2, v_3 be the columns of U, with*

$$v_j = \frac{1}{\sqrt{2}^k} \begin{pmatrix} a_j + b_j\sqrt{2} \\ c_j + d_j\sqrt{2} \\ e_j + f_j\sqrt{2} \end{pmatrix},$$

for $a_j, \ldots, f_j \in \mathbb{Z}$. Then for all $j, \ell \in \{1, 2, 3\}$,

$$a_j b_\ell + b_j a_\ell + c_j d_\ell + d_j c_\ell + e_j f_\ell + f_j e_\ell = 0 \tag{6.16}$$

and

$$a_j a_\ell + c_j c_\ell + e_j e_\ell + 2(b_j b_\ell + d_j d_\ell + f_j f_\ell) = 2^k \langle v_j, v_\ell \rangle. \tag{6.17}$$

In particular, we have, for all $j \in \{1, 2, 3\}$,

$$a_j b_j + c_j d_j + e_j f_j = 0 \tag{6.18}$$

and

$$a_j^2 + c_j^2 + e_j^2 + 2(b_j^2 + d_j^2 + f_j^2) = 2^k. \tag{6.19}$$

Proof. Computing the inner product, we have

$$\langle v_j, v_\ell \rangle = \frac{1}{2^k} \left(a_j a_\ell + c_j c_\ell + e_j e_\ell + 2(b_j b_\ell + d_j d_\ell + f_j f_\ell) + \sqrt{2}(a_j b_\ell + b_j a_\ell + c_j d_\ell + d_j c_\ell + e_j f_\ell + f_j e_\ell) \right).$$
$$\tag{6.20}$$

Since $U^\dagger U = I$, we have $\langle v_j, v_j \rangle = 1$, and $\langle v_j, v_\ell \rangle = 0$ when $\ell \neq j$. Therefore, the coefficient of $\sqrt{2}$ in equation equation (6.20) on the following page must be zero, proving equation (6.16) on the next page and equation (6.17) on the following page. Equations equation (6.18) and equation (6.19) immediately follow by letting $j = \ell$. $\qquad\square$

Remark 6.3.18. In Lemma 6.3.17, we have worked with columns v_j of the matrix U. But since U is orthogonal, the analogous properties also hold for the rows of U.

Lemma 6.3.19. *Let $U \in SO(3)$ be an orthogonal matrix with entries in $\mathbb{D}[\sqrt{2}]$, and with least denominator exponent $k = 0$. Then U the Bloch sphere representation of some Clifford operator.*

Proof. Let v_j be any column of U, with the notation of Lemma 6.3.17. By equation (6.19), we have $a_j^2 + c_j^2 + e_j^2 + 2(b_j^2 + d_j^2 + f_j^2) = 1$. Since each summand is a positive integer, we must have $b_j, d_j, f_j = 0$, and exactly one of a_j, c_j or $e_j = \pm 1$, for each $j = 1, 2, 3$. Therefore, all the matrix entries are in $\{-1, 0, 1\}$, and the claim follows by Remark 6.3.8 on page 128. $\square$

Lemma 6.3.20. *Let $U \in SO(3)$ be an orthogonal matrix with entries in $\mathbb{D}[\sqrt{2}]$, and let k be the least denominator exponent of U. If $k = 0$, then $p_k(U) \sim_{\mathbb{c}} M_1$. If $k > 0$, then $p_k(U) \sim_{\mathbb{c}} M$ for some $M \in \{M_T, M_H, M_S\}$, where*

$$M_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad M_T = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad M_H = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix}, \quad M_S = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix}.$$

Proof. First consider the case $k = 0$. Let v_j be any column of U, with the notation of Lemma 6.3.17 on the previous page. By equation (6.19) on the preceding page, we have $a_j^2 + c_j^2 + e_j^2 + 2(b_j^2 + d_j^2 + f_j^2) = 1$. Since each summand is a positive integer, we must have $b_j, d_j, f_j = 0$, and exactly one of a_j, c_j or $e_j = \pm 1$, for each $j = 1, 2, 3$. Noting that the columns of U are orthogonal, we see that $p_k(U) \sim_{\mathbb{c}} M_1$.

Now consider the case $k > 0$. Let v_j be any row or column of U, with the notation of Lemma 6.3.17 on the previous page. By equation (6.19) on the preceding page, it follows that $a_j^2 + c_j^2 + e_j^2$ is even, and therefore an even number of a_j, c_j, and e_j have parity 1. Therefore, each row or column of $p_k(U)$ has an even number of 1's. Moreover, since k is the least denominator exponent of U, $p_k(U)$ has at least one non-zero entry. Modulo a permutation of columns, this leaves exactly four possibilities for $p_k(U)$:

$$(a) \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad (b) \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix}, \quad (c) \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix}, \quad (d) \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}.$$

In cases (a)–(c), we are done. Case (d) is impossible because it implies that $a_1 a_2 + c_1 c_2 + e_1 e_2$ is odd, contradicting the fact that it is even by . $\qquad\square$

Lemma 6.3.21. *Let $U \in SO(3)$ be an orthogonal matrix with entries in $\mathbb{D}[\sqrt{2}]$, and with least denominator exponent $k > 0$. Then there exists $N \in \{T, HT, SHT\}$ such that the least denominator exponent of $\hat{N}^{-1}U$ is $k - 1$.*

Proof. By Lemma 6.3.20, we know that $p_k(U) \sim_{\mathbb{C}} M$, for some $M \in \{M_T, M_H, M_S\}$. We consider each of these cases.

1. $p_k(U) \sim_{\mathbb{C}} M_T$. By assumption, U has two columns v with $p_k(v) = (1, 1, 0)^T$.

   Let
   $$
   v = \frac{1}{\sqrt{2}^k} \begin{pmatrix} a + b\sqrt{2} \\ c + d\sqrt{2} \\ e + f\sqrt{2} \end{pmatrix}
   $$
   be any such column. By equation (6.18) on the previous page, we have $ab + cd + ef = 0$. Since $\bar{e} = 0$, we have $\bar{a}\bar{b} + \bar{c}\bar{d} = 0$. Since $\bar{a} = \bar{c} = 1$, we can conclude $\bar{b} + \bar{d} = 0$. Applying $\hat{T}^{-1}$ to v, we compute:
   $$
   \hat{T}^{-1}v = \frac{1}{\sqrt{2}^{k+1}} \begin{bmatrix} c + a & + & (d+b)\sqrt{2} \\ c - a & + & (d-b)\sqrt{2} \\ e\sqrt{2} & + & 2f \end{bmatrix} = \frac{1}{\sqrt{2}^{k-1}} \begin{bmatrix} \frac{c+a}{2} & + & \frac{d+b}{\sqrt{2}} \\ \frac{c-a}{2} & + & \frac{d-b}{\sqrt{2}} \\ \frac{e}{\sqrt{2}} & + & f \end{bmatrix} = \frac{1}{\sqrt{2}^{k-1}} \begin{bmatrix} a' & + & b'\sqrt{2} \\ c' & + & d'\sqrt{2} \\ f & + & e'\sqrt{2} \end{bmatrix}
   $$
   where $a' = \frac{c+a}{2}, b' = \frac{d+b}{2}, c' = \frac{c-a}{2}, d' = \frac{d-b}{2}$ and $e' = \frac{e}{2}$ are all integers. Hence, $k - 1$ is a denominator exponent of $\hat{T}^{-1}v$. Moreover, since $a' + c' = c$ is odd, one of a' and c' is odd, proving that $k - 1$ is the least denominator exponent of $\hat{T}^{-1}v$.

   Now consider the third column w of U, where $p_k(w) = (0, 0, 0)^T$. Then $k - 1$ is a denominator exponent of w, so that k is a denominator exponent for $\hat{T}^{-1}w$.

   Let
   $$
   p_k(\hat{T}^{-1}w) = \begin{pmatrix} x \\ y \\ z \end{pmatrix}.
   $$

As the least denominator exponent of the other two column of $p_k(\hat{T}^{-1}U)$ is $k-1$, we have

$$p_k(\hat{T}^{-1}U) \sim_{\mathfrak{e}} \begin{bmatrix} 0 & 0 & x \\ 0 & 0 & y \\ 0 & 0 & z \end{bmatrix}.$$

But $\hat{T}^{-1}U$ is orthogonal, so by equation (6.19) on page 133, applied to each row of $\hat{T}^{-1}U$, we conclude that $x = y = z = 0$. It follows that the least denominator exponent of $\hat{T}^{-1}U$ is $k-1$.

2. $p_k(U) \sim_{\mathfrak{e}} M_H$. In this case, $p_k(\hat{H}^{-1}U) \sim_{\mathfrak{e}} p(\hat{H}^{-1}M_H) = M_T$. We then continue as in case 1.

3. $p_k(U) \sim_{\mathfrak{e}} M_S$. In this case, $p_k(\hat{H}^{-1}\hat{S}^{-1}U) \sim_{\mathfrak{e}} p(\hat{H}^{-1}\hat{S}^{-1}M_S) = M_T$. We then continue as in case 1. □

Combining Lemmas 6.3.19 on page 134 and 6.3.20 on page 134, we easily get the following result:

Theorem 6.3.22. *Let $U \in SO(3)$ be an orthogonal matrix. Then U is the Bloch sphere representation of some Clifford+T operator M if and only if the entries of U are in the ring $\mathbb{D}[\sqrt{2}]$.*

Proof. The "only if" direction is trivial, since all the generators of the Clifford+T group have this property (see equation (6.15)). To prove the "if" direction, let k be the least denominator exponent of U. We proceed by induction on k. If $k = 0$, by Lemma 6.3.19, U is the Bloch sphere representation of some Clifford operator, and therefore a Clifford+T operator. If $k > 0$, then by Lemma 6.3.20, we can write $U = \hat{N}U'$, where $N \in \{T, HT, SHT\}$ and U' has least denominator exponent $k-1$. By induction hypothesis, U' is a Clifford+T operator, and therefore so is U. □

Remark 6.3.23. Combining this result with the algorithm of Theorem 6.3.16 on page 132, we have a linear-time algorithm for computing the Matsumoto-Amano normal form of any unitary operator $U \in SO(3)$ with entries in $\mathbb{D}[\sqrt{2}]$.

Corollary 6.3.24 (Kliuchnikov et al. [12])**.** *Let $U \in U(2)$ be a unitary matrix. Then U is a Clifford+T operator if and only if the matrix entries of U are in the ring $\mathbb{D}[\sqrt{2}, i]$.*

Proof. Again, the "only if" direction is trivial, as it is true for the generators. For the "if" direction, it suffices to note that, by equation (6.14) on page 128, whenever U takes its entries in $\mathbb{D}[\sqrt{2}, i]$, then $\hat{U}$ takes its entries in $\mathbb{D}[\sqrt{2}]$.[4]   □

Corollary 6.3.24 on the next page was first proved by Kliuchnikov et al. [12], using a direct method (i.e., not going via the Bloch sphere representation). It is interesting to note that Theorem 6.3.22 on the preceding page is stronger than Corollary 6.3.24, in the sense that the Theorem obviously implies the Corollary, whereas it is not a priori obvious that the Corollary implies the Theorem.

6.3.6 Alternative normal forms

With the exception of the left-most and right-most gates, the Matsumoto-Amano normal form uses syllables of the form HT and SHT. It is of course possible to use different sets of syllables instead.

E-T normal form

Consider the Clifford operator

$$E = HS^3 \omega^3 = \frac{1}{2} \begin{pmatrix} -1 + i & 1 + i \\ -1 + i & -1 - i \end{pmatrix}.$$

It has the following properties:

$$E^3 = I, \quad EXE^{-1} = Y, \quad EYE^{-1} = Z, \quad EZE^{-1} = X.$$

---

[4]Todo: treat phase correctly; also introduce the ring $\mathbb{D}[\sqrt{2}, i]$ at some appropriate time

The operator E serves as a convenient operator for switching between the X-, Y-, and Z-bases. On the Bloch sphere, it represents a rotation by 120 degrees about the axis $(1, 1, 1)^T$:

$$\hat{E} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

The operators E and E^2 have properties analogous to H and SH. Specifically, if we let $\mathcal{H} = \{I, E, E^2\}$ and $\mathcal{H}' = \{E, E^2\}$, then the properties of Lemma 6.3.3 are satisfied. The proofs of Proposition 6.3.4 and Corollary 6.3.5 only depend on these properties, and the uniqueness proof (Theorem 6.3.7 on page 128) also goes through without significant changes. We therefore have:

Proposition 6.3.25 (*E-T normal form*)**.** *Every single-qubit Clifford+T operator can be uniquely written in the form*

$$(T \mid \varepsilon) \, (ET \mid E^2T)^* \, \mathcal{C}. \tag{6.21}$$

Moreover, this normal form has minimal T-count, and there exists a linear-time algorithm for symbolically reducing any sequence of Clifford+T operators to this normal form.

T_x-T_y-T_z normal form

It is plain to see that every syllable of the E-T normal form (except perhaps the first or last one) consists of a 45 degree z-rotation, followed by a basis change that rotates either the x- or y-axis into the z-position. Abstracting away from these basis changes, the entire normal form can therefore be regarded as a sequence of 45-degree rotations about the x-, y-, and z-axes. More precisely, let us define variants of the T-gate that rotate about the three different axes:

$$T_x = ETE^2,$$
$$T_y = E^2TE,$$
$$T_z = T.$$

Using the commutativities $ET_x = T_yE$, $ET_y = T_zE$, and $ET_z = T_xE$, it is then clear that every expression of the form (equation (6.21) on the next page) can be uniquely rewritten

as a sequence of T_x, T_y, and T_z rotations, with no repeated symbol, followed by a Clifford operator. This can be easily proved by induction, but is best seen in an example:

$$
\begin{aligned}
TETETE^2TEC &= T_zET_zET_zE^2T_zEC \\
&\rightarrow T_zT_xE^2T_zE^2T_zEC \\
&\rightarrow T_zT_xT_yE^4T_zEC \\
&\rightarrow T_zT_xT_yET_zEC \\
&\rightarrow T_zT_xT_yT_xE^2C \\
&\rightarrow T_zT_xT_yT_xC'.
\end{aligned}
$$

We have:

Proposition 6.3.26 (T_x-T_y-T_z normal form). *Every single-qubit Clifford+T operator can be uniquely written in the form*

$$T_{r_1}T_{r_2}\ldots T_{r_n}C,$$

where $n \geq 0$, $r_1, \ldots, r_n \in \{x, y, z\}$, and $r_i \neq r_{i+1}$ for all $i \leq n-1$. We define the T-count of such an expression to be n; then this normal form has minimal T-count. Moreover, there exists a linear-time algorithm for symbolically reducing any sequence of Clifford+T operators to this normal form.

The T_x-T_y-T_z normal form is, in a sense, the most "canonical" one of the normal forms considered here; it also explains why T-count is an appropriate measure of the size of a Clifford+T operator. In a physical quantum computer with error correction, there is in general no reason to expect the T_z gate to be more privileged than the T_x or T_y gates; one may imagine that it would be efficient for a quantum computer to provide all three T-gates as primitive logical operations.

Bocharov-Svore normal form

Bocharov and Svore [8, Prop.1] consider the following normal form for single-qubit Clifford+T circuits:

$$(H \mid \varepsilon)(TH \mid SHTH)^* \mathcal{C}. \tag{6.22}$$

This normal form is not unique; for example, $H.H$ and I are two different normal forms denoting the same operator, as are $SHTH.Z$ and $H.SHTH$. (Here we have used a dot to delimit syllables; this is for readability only). Recall that two regular expressions are *equivalent* if they define the same set of strings. Using laws of regular expressions, we can equivalently rewrite (equation (6.22) on page 140) as

$$((\epsilon \mid T \mid SHT)(HT \mid HSHT)^* H\mathcal{C}) \mid \mathcal{C}. \tag{6.23}$$

Since $H\mathcal{C}$ is just a redundant way to write a Clifford operator, we can simplify it to $\mathcal{C}$; moreover, in this case, $\epsilon\mathcal{C}$ and $\mathcal{C}$ are the same, so (equation (6.23) on page 140) simplifies to

$$(\epsilon \mid T \mid SHT)(HT \mid HSHT)^* \mathcal{C}. \tag{6.24}$$

Moreover, since $SHT = HSHT.X$, any expression starting with SHT can be rewritten as one starting with $HSHT$, so the SHT syllable is redundant and we can eliminate it:

$$(\epsilon \mid T)(HT \mid HSHT)^* \mathcal{C}. \tag{6.25}$$

Let us say that an operator is in *Bocharov-Svore normal form* if it is written in the form (equation (6.25) on the following page). This version of the Bocharov-Svore normal form is indeed unique; note that it is almost the same as the Matsumoto-Amano normal form, except that the syllable SHT has been replaced by $HSHT$. Since the set $\mathcal{H} = \{I, H, HSH\}$ satisfies Lemma 6.3.3 on page 125, existence, uniqueness, T-optimality, and efficiency are proved in the same way as for the Matsumoto-Amano and E-T normal forms.

Bocharov and Svore [8, Prop.2] also consider a second normal form, which has Clifford operators on both sides, but the first four interior syllables restricted to TH:

$$\mathcal{C} \left(\epsilon \mid TH \mid (TH)^2 \mid (TH)^3 \mid (TH)^4 (TH \mid SHTH)^* \right) \mathcal{C} \tag{6.26}$$

However, this normal form is not at all unique; for instance, $Z.TH$ and $TH.X$ denote the same operator, as do $YS.TH.TH$ and $TH.TH.X\omega$.

6.3.7 Matsumoto-Amano normal forms and U(2)

Example 6.3.27. Consider the matrix

$$U = \frac{1}{\sqrt{2}^3} \begin{pmatrix} \omega^2 + \omega & -2\omega^3 + \omega^2 + \omega \\ \omega^3 - 2\omega^2 - 1 & -\omega^3 + 1 \end{pmatrix}.$$

It has least denominator exponent 3. Its 3-, 4-, and 5-residues are:

$$\rho_3(U) = \begin{pmatrix} 0110 & 0110 \\ 1001 & 1001 \end{pmatrix}, \quad \rho_4(U) = \begin{pmatrix} 1111 & 1111 \\ 1111 & 1111 \end{pmatrix}, \quad \rho_5(U) = 0.$$

Definition 6.3.28. Let $\mathcal{S}^\omega$ be the 64-element subgroup of the Clifford group in U(2) spanned by S, X and ω. Then $\sim_{\mathcal{S}^\omega}$ defined by right multiplication by $\mathcal{S}^\omega$ is an equivalence relation on the residue matrices of operators in the Clifford+T group.

Remark 6.3.29. The equivalence relation $\sim_{\mathcal{S}^\omega}$ is characterized by the following operations:

1. "Rotating" all of the entries in the matrix by 1,2 or 3 positions. This corresponds to multiplication by a power of ω.

2. Swapping the two columns. The corresponds to the right action of X.

3. "Rotating" the entries of the second column by two positions. This corresponds to the right action of S.

Remark 6.3.30. The residue matrices in figure figure 6.2 on page 142 are modulo $\sim_{\mathcal{S}^\omega}$.

Lemma 6.3.31. *Let $k \geq 2$ and $b, d \in \mathbb{Z}[\omega]$. Then $\frac{1}{\sqrt{2}^k} \begin{pmatrix} 1 + 2b \\ 1 + 2d \end{pmatrix}$ is not a unit vector.*

Proof. Suppose otherwise, then

$$2^k = 1 + 2(b + b^t) + 4bb^t +$$

$$1 + 2(d + d^t) + 4dd^t,$$

so $2 = 2^k - 2(b + b^t) - 2(d + d^t) - 4(bb^t + dd^t)$. By lemma 6.2.16 on page 122, the right hand side of this is divisible in $\mathbb{Z}[\omega]$ by $2\sqrt{2}$, while the left hand side is not. Thus we have a contradiction. □
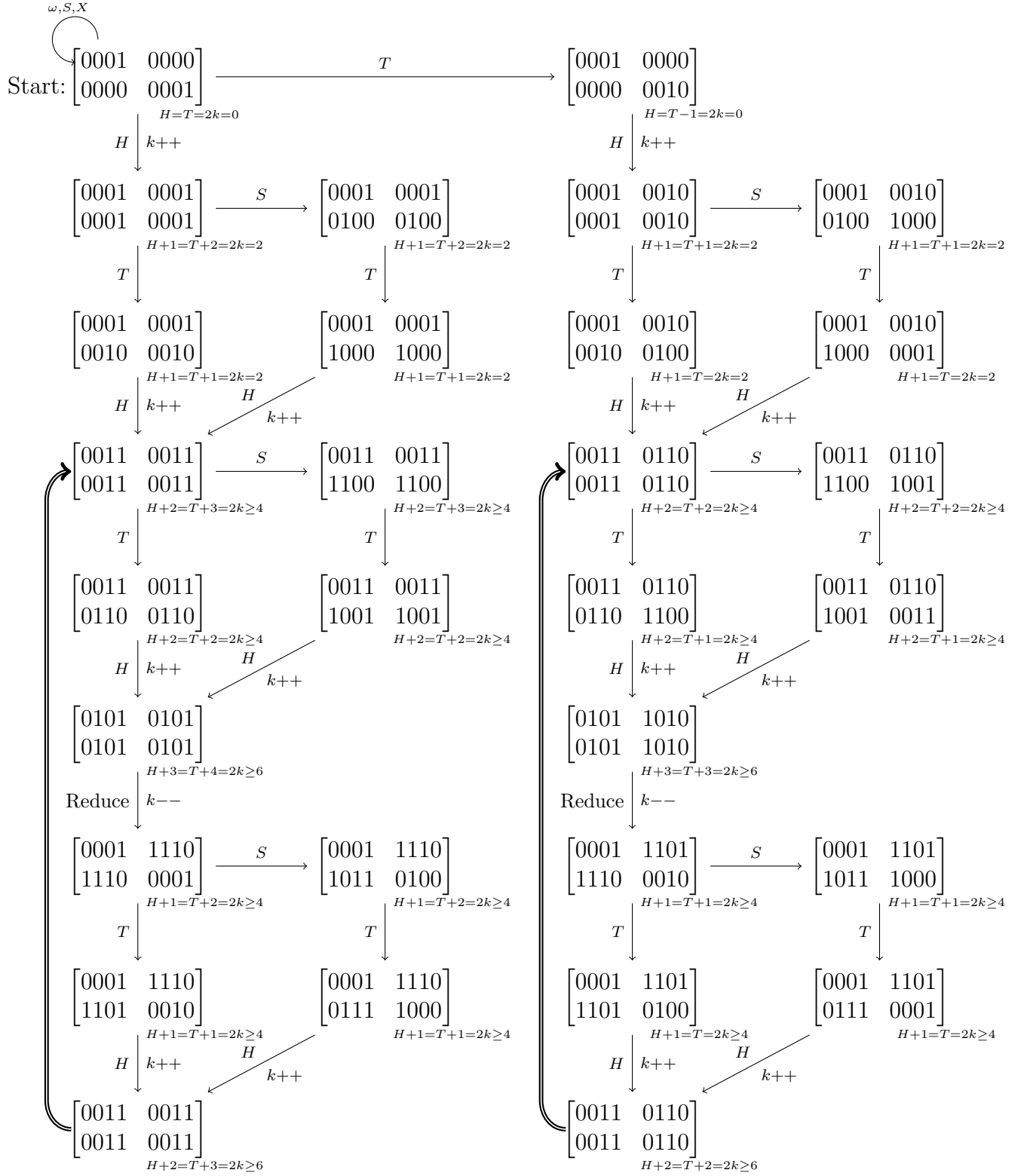
Start: $\begin{bmatrix} 0001 & 0000 \\ 0000 & 0001 \end{bmatrix}$ $\xrightarrow{\quad\quad T \quad\quad}$ $\begin{bmatrix} 0001 & 0000 \\ 0000 & 0010 \end{bmatrix}$

(self-loop: ω,S,X)

$H=T=2k=0$ $\qquad\qquad\qquad\qquad\qquad H=T-1=2k=0$

$H \downarrow k{+}{+}$ $\qquad\qquad\qquad\qquad\qquad\qquad H \downarrow k{+}{+}$

$\begin{bmatrix} 0001 & 0001 \\ 0001 & 0001 \end{bmatrix} \xrightarrow{S} \begin{bmatrix} 0001 & 0001 \\ 0100 & 0100 \end{bmatrix}$ $\qquad$ $\begin{bmatrix} 0001 & 0010 \\ 0001 & 0010 \end{bmatrix} \xrightarrow{S} \begin{bmatrix} 0001 & 0010 \\ 0100 & 1000 \end{bmatrix}$

$H{+}1{=}T{+}2{=}2k{=}2$ $\quad$ $H{+}1{=}T{+}2{=}2k{=}2$ $\qquad$ $H{+}1{=}T{+}1{=}2k{=}2$ $\quad$ $H{+}1{=}T{+}1{=}2k{=}2$

$T \downarrow$ $\qquad\qquad\quad T \downarrow$ $\qquad\qquad\qquad T \downarrow$ $\qquad\qquad\quad T \downarrow$

$\begin{bmatrix} 0001 & 0001 \\ 0010 & 0010 \end{bmatrix}$ $\begin{bmatrix} 0001 & 0001 \\ 1000 & 1000 \end{bmatrix}$ $\qquad$ $\begin{bmatrix} 0001 & 0010 \\ 0010 & 0100 \end{bmatrix}$ $\begin{bmatrix} 0001 & 0010 \\ 1000 & 0001 \end{bmatrix}$

$H{+}1{=}T{+}1{=}2k{=}2$ $\quad$ $H{+}1{=}T{+}1{=}2k{=}2$ $\qquad$ $H{+}1{=}T{=}2k{=}2$ $\quad$ $H{+}1{=}T{=}2k{=}2$

$H \downarrow k{+}{+}$ $\quad$ (H, $k{+}{+}$ diag) $\qquad\qquad H \downarrow k{+}{+}$ $\quad$ (H, $k{+}{+}$ diag)

$\begin{bmatrix} 0011 & 0011 \\ 0011 & 0011 \end{bmatrix} \xrightarrow{S} \begin{bmatrix} 0011 & 0011 \\ 1100 & 1100 \end{bmatrix}$ $\qquad$ $\begin{bmatrix} 0011 & 0110 \\ 0011 & 0110 \end{bmatrix} \xrightarrow{S} \begin{bmatrix} 0011 & 0110 \\ 1100 & 1001 \end{bmatrix}$

$H{+}2{=}T{+}3{=}2k{\geq}4$ $\quad$ $H{+}2{=}T{+}3{=}2k{\geq}4$ $\qquad$ $H{+}2{=}T{+}2{=}2k{\geq}4$ $\quad$ $H{+}2{=}T{+}2{=}2k{\geq}4$

$T \downarrow$ $\qquad\qquad\quad T \downarrow$ $\qquad\qquad\qquad T \downarrow$ $\qquad\qquad\quad T \downarrow$

$\begin{bmatrix} 0011 & 0011 \\ 0110 & 0110 \end{bmatrix}$ $\begin{bmatrix} 0011 & 0011 \\ 1001 & 1001 \end{bmatrix}$ $\qquad$ $\begin{bmatrix} 0011 & 0110 \\ 0110 & 1100 \end{bmatrix}$ $\begin{bmatrix} 0011 & 0110 \\ 1001 & 0011 \end{bmatrix}$

$H{+}2{=}T{+}2{=}2k{\geq}4$ $\quad$ $H{+}2{=}T{+}2{=}2k{\geq}4$ $\qquad$ $H{+}2{=}T{+}1{=}2k{\geq}4$ $\quad$ $H{+}2{=}T{+}1{=}2k{\geq}4$

$H \downarrow k{+}{+}$ $\quad$ (H, $k{+}{+}$ diag) $\qquad\qquad H \downarrow k{+}{+}$ $\quad$ (H, $k{+}{+}$ diag)

$\begin{bmatrix} 0101 & 0101 \\ 0101 & 0101 \end{bmatrix}$ $\qquad\qquad\qquad\qquad$ $\begin{bmatrix} 0101 & 1010 \\ 0101 & 1010 \end{bmatrix}$

$H{+}3{=}T{+}4{=}2k{\geq}6$ $\qquad\qquad\qquad$ $H{+}3{=}T{+}3{=}2k{\geq}6$

Reduce $\downarrow k{-}{-}$ $\qquad\qquad\qquad\qquad$ Reduce $\downarrow k{-}{-}$

$\begin{bmatrix} 0001 & 1110 \\ 1110 & 0001 \end{bmatrix} \xrightarrow{S} \begin{bmatrix} 0001 & 1110 \\ 1011 & 0100 \end{bmatrix}$ $\qquad$ $\begin{bmatrix} 0001 & 1101 \\ 1110 & 0010 \end{bmatrix} \xrightarrow{S} \begin{bmatrix} 0001 & 1101 \\ 1011 & 1000 \end{bmatrix}$

$H{+}1{=}T{+}2{=}2k{\geq}4$ $\quad$ $H{+}1{=}T{+}2{=}2k{\geq}4$ $\qquad$ $H{+}1{=}T{+}1{=}2k{\geq}4$ $\quad$ $H{+}1{=}T{+}1{=}2k{\geq}4$

$T \downarrow$ $\qquad\qquad\quad T \downarrow$ $\qquad\qquad\qquad T \downarrow$ $\qquad\qquad\quad T \downarrow$

$\begin{bmatrix} 0001 & 1110 \\ 1101 & 0010 \end{bmatrix}$ $\begin{bmatrix} 0001 & 1110 \\ 0111 & 1000 \end{bmatrix}$ $\qquad$ $\begin{bmatrix} 0001 & 1101 \\ 1101 & 0100 \end{bmatrix}$ $\begin{bmatrix} 0001 & 1101 \\ 0111 & 0001 \end{bmatrix}$

$H{+}1{=}T{+}1{=}2k{\geq}4$ $\quad$ $H{+}1{=}T{+}1{=}2k{\geq}4$ $\qquad$ $H{+}1{=}T{=}2k{\geq}4$ $\quad$ $H{+}1{=}T{=}2k{\geq}4$

$H \downarrow k{+}{+}$ $\quad$ (H, $k{+}{+}$ diag) $\qquad\qquad H \downarrow k{+}{+}$ $\quad$ (H, $k{+}{+}$ diag)

$\begin{bmatrix} 0011 & 0011 \\ 0011 & 0011 \end{bmatrix}$ $\qquad\qquad\qquad\qquad$ $\begin{bmatrix} 0011 & 0110 \\ 0011 & 0110 \end{bmatrix}$

$H{+}2{=}T{+}3{=}2k{\geq}6$ $\qquad\qquad\qquad$ $H{+}2{=}T{+}2{=}2k{\geq}6$

Figure 6.2: Transitions of residue matrices in U2 when applying the Matsumoto-Amano algorithm

Lemma 6.3.32. *Given a unitary matrix $U \in \mathbb{D}[\omega]^{2\times 2}$ with least denominator exponent $k \geq 2$, such that:*

1. $\rho_{k+1}(U) = \begin{bmatrix} 0101 & 0101 \\ 0101 & 0101 \end{bmatrix}$ *and*

2. $\rho_k(HU) = \begin{bmatrix} 0011 & 0011 \\ 0110 & 0110 \end{bmatrix}$,

then, $\rho_k(U) = \begin{bmatrix} 0010 & 1101 \\ 1101 & 0010 \end{bmatrix}$

Proof. Referencing Table table 6.1 on page 121, we see the first condition limits the possible choices for the entries of $\rho_k(U)$ to the set $\{0010, 0111, 1000, 1101\}$. The second condition implies that $\rho_{k+1}(HU)$ is reducible and in fact that each entry is 1111. This means each column of U must be either $[0010, 1101]^t, [1101, 0010]^t, [0111, 1000]^t$ or $[1000, 0111]^t$. As we are considering equivalence classes, we can assume without loss of generality, that the columns are in $\{[0010, 1101]^t, [1101, 0010]^t\}$. But by Lemma 6.3.31, we can not have a row like $[0010, 0010]$, therefore $U = \begin{bmatrix} 0010 & 1101 \\ 1101 & 0010 \end{bmatrix}$. $\qquad\square$

Corollary 6.3.33. *In Figure figure 6.2 on the preceding page, the transitions labelled with "Reduce" are correct.*

Proof. For the left most reduce, we directly apply Lemma 6.3.32 and then note that

$$\begin{bmatrix} 0001 & 1110 \\ 1110 & 0001 \end{bmatrix} \sim_{s\omega} \begin{bmatrix} 0010 & 1101 \\ 1101 & 0010 \end{bmatrix}.$$

For the rightmost reduce, the same argument as shown in Lemma 6.3.32 on the preceding page can be applied to the preconditions of this reduce, giving us a similar result. $\qquad\square$

6.4 Exact synthesis of multi-qubit operators

Here, we focus on the problem of exact synthesis for n-qubit operators, using the Clifford+T universal gate set. Recall that the Clifford group on n qubits is generated by the Hadamard gate H, the phase gate S, the controlled-not gate, and the scalar $\omega = e^{i\pi/4}$ (one may allow arbitrary unit scalars, but it is not convenient for our purposes to do so). It is well-known that one obtains a universal gate set by adding the non-Clifford operator T [17].

$$\omega = e^{i\pi/4}, \quad H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix},$$

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}. \tag{6.27}$$

In addition to the Clifford+T group on n qubits, as defined above, we also consider the slightly larger group of Clifford+T operators "with ancillas". We say that an n-qubit operator U is a Clifford+T operator *with ancillas* if there exists $m \geq 0$ and a Clifford+T operator U' on $n + m$ qubits, such that $U'(\ket{\phi} \otimes \ket{0}) = (U\ket{\phi}) \otimes \ket{0}$ for all n-qubit states $\ket{\phi}$.

Kliuchnikov, Maslov, and Mosca [12] showed that a single-qubit operator U is in the Clifford+T group if and only if all of its matrix entries belong to the ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$. They also showed that the Clifford+T groups "with ancillas" and "without ancillas" coincide for $n = 1$, but not for $n \geq 2$. Moreover, Kliuchnikov et al. conjectured that for all n, an n-qubit operator U is in the Clifford+T group with ancillas if and only if its matrix entries belong to $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$. They also conjectured that a single ancilla qubit is always sufficient in the representation of a Clifford+T operator with ancillas. This section of the thesis will prove these conjectures. In particular, this yields an algorithm for exact Clifford+T synthesis of n-qubit operators. We also obtain a characterization of the Clifford+T group on n qubits

without ancillas.

It is important to note that, unlike in the single-qubit case, the circuit synthesized here are not in any sense canonical, and very far from optimal. Thus, the question of *efficient* synthesis is not addressed here.

6.4.1   Decomposition into two-level matrices

Recall that a *two-level matrix* is an $n \times n$-matrix that acts non-trivially on at most two vector components [17]. If

$$U = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

is a 2×2-matrix and $j \neq \ell$, we write $U_{[j,\ell]}$ for the two-level $n \times n$-matrix defined by

$$
U_{[j,\ell]} =
\begin{array}{c}
\\
\vdots \\
j \\
\vdots \\
\ell \\
\vdots
\end{array}
\begin{array}{c}
\cdots\quad j\quad\cdots\quad \ell\quad\cdots \\
\left(
\begin{array}{ccccc}
I & & & & \\
 & a & & b & \\
 & & I & & \\
 & c & & d & \\
 & & & & I
\end{array}
\right),
\end{array}
$$

and we say that $U_{[j,\ell]}$ is a two-level matrix *of type U*. Similarly, if a is a scalar, we write $a_{[j]}$ for the one-level matrix

$$
a_{[j]} =
\begin{array}{c}
\\
\vdots \\
j \\
\vdots
\end{array}
\begin{array}{c}
\cdots\quad j\quad\cdots \\
\left(
\begin{array}{ccc}
I & & \\
 & a & \\
 & & I
\end{array}
\right),
\end{array}
$$

and we say that $a_{[j]}$ is a one-level matrix *of type a*.

Lemma 6.4.1 (Row operation). *Let $u = (u_1, u_2)^T \in \mathbb{D}[\omega]^2$ be a vector with denominator exponent $k > 0$ and k-residue $\rho_k(u) = (x_1, x_2)$, such that $x_1^\dagger x_1 = x_2^\dagger x_2$. Then there exists*

145

a sequence of matrices $U_1, \ldots, U_h$, each of which is H or T, such that $v = U_1 \cdots U_h u$ has denominator exponent $k - 1$, or equivalently, $\rho_k(v)$ is defined and reducible.

Proof. It can be seen from Table table 6.1 on page 121 that $x_1^\dagger x_1$ is either 0000, 1010, or 0001.

- Case 1: $x_1^\dagger x_1 = x_2^\dagger x_2 = 0000$. In this case, $\rho_k(u)$ is already reducible, and there is nothing to show.

- Case 2: $x_1^\dagger x_1 = x_2^\dagger x_2 = 1010$. In this case, we know from Table table 6.1 on page 121 that $x_1, x_2 \in \{0011, 0110, 1100, 1001\}$. In particular, x_1 is a cyclic permutation of x_2, say, $x_1 = \omega^m x_2$. Let $v = HT^m u$. Then

$$\rho_k(\sqrt{2}\, v) = \rho_k \left(\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \omega^m \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \right)$$

$$= \rho_k \begin{pmatrix} u_1 + \omega^m u_2 \\ u_1 - \omega^m u_2 \end{pmatrix}$$

$$= \begin{pmatrix} x_1 + \omega^m x_2 \\ x_1 - \omega^m x_2 \end{pmatrix} = \begin{pmatrix} 0000 \\ 0000 \end{pmatrix}.$$

  This shows that $\rho_k(\sqrt{2}\, v)$ is twice reducible; therefore, $\rho_k(v)$ is defined and reducible as claimed.

- Case 3: $x_1^\dagger x_1 = x_2^\dagger x_2 = 0001$. In this case, we know from Table table 6.1 on page 121 that $x_1, x_2 \in \{0001, 0010, 0100, 1000\} \cup \{0111, 1110, 1101, 1011\}$. If both x_1, x_2 are in the first set, or both are in the second set, then x_1 and x_2 are cyclic permutations of each other, and we proceed as in case 2. The only remaining cases are that x_1 is a cyclic permutation of 0001 and x_2 is a cyclic permutation of 0111, or vice versa. But then there exists some m such that

$x_1 + \omega^m x_2 = 1111$. Letting $u' = HT^m u$, we have

$$\rho_k(\sqrt{2}\, u') = \rho_k(\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \omega^m \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix})$$

$$= \rho_k \begin{pmatrix} u_1 + \omega^m u_2 \\ u_1 - \omega^m u_2 \end{pmatrix}$$

$$= \begin{pmatrix} x_1 + \omega^m x_2 \\ x_1 - \omega^m x_2 \end{pmatrix} = \begin{pmatrix} 1111 \\ 1111 \end{pmatrix}.$$

Since this is reducible, u' has denominator exponent k. Let $\rho_k(u') = (y_1, y_2)$. Because $\sqrt{2}\, y_1 = \sqrt{2}\, y_2 = 1111$, we see from Table table 6.1 on page 121 that $y_1, y_2 \in \{0011, 0110, 1100, 1001\}$ and $y_1^\dagger y_1 = y_2^\dagger y_2 = 1010$. Therefore, u' satisfies the condition of case 2 above. Proceeding as in case 2, we find m' such that $v = HT^{m'} u' = HT^{m'} HT^m u$ has denominator exponent $k - 1$. This finishes the proof. $\qquad \square$

Lemma 6.4.2 (Column lemma). *Consider a unit vector $u \in \mathbb{D}[\omega]^n$, i.e., an n-dimensional column vector of norm 1 with entries from the ring $\mathbb{D}[\omega]$. Then there exist a sequence $U_1, \ldots, U_h$ of one- and two-level unitary matrices of types X, H, T, and ω such that $U_1 \cdots U_h u = e_1$, the first standard basis vector.*

Proof. The proof is by induction on k, the least denominator exponent of u. Let $u = (u_1, \ldots, u_n)^T$.

- Base case. Suppose $k = 0$. Then $u \in \mathbb{Z}[\omega]^n$. Since by assumption $\|u\|^2 = 1$, it follows by Lemma 6.2.6 on page 120 that $\|u\|_{\text{weight}}^2 = 1$. Since $u_1, \ldots, u_n$ are elements of $\mathbb{Z}[\omega]$, their weights are non-negative integers. It follows that there is precisely one j with $\|u_j\|_{\text{weight}} = 1$, and $\|u_\ell\|_{\text{weight}} = 0$ for all $\ell \neq j$. Let $u' = X_{[1,j]} u$ if $j \neq 1$, and $u' = u$ otherwise. Now u'_1 is of the form ω^{-m}, for some $m \in \{0, \ldots, 7\}$, and $u'_\ell = 0$ for all $\ell \neq 1$. We have $\omega_{[1]}^m u' = e_1$, as desired.

- Induction step. Suppose $k > 0$. Let $v = \sqrt{2}^k u \in \mathbb{Z}[\omega]^n$, and let $x = \rho_k(u) = \rho(v)$. From $\|u\|^2 = 1$, it follows that $\|v\|^2 = v_1^\dagger v_1 + \ldots + v_n^\dagger v_n = 2^k$. Taking

147

residues of the last equation, we have

$$x_1^\dagger x_1 + \ldots + x_n^\dagger x_n = 0000. \tag{6.28}$$

It can be seen from Table table 6.1 on page 121 that each summand $x_j^\dagger x_j$ is either 0000, 0001, or 1010. Since their sum is 0000, it follows that there is an even number of j such that $x_j^\dagger x_j = 0001$, and an even number of j such that $x_j^\dagger x_j = 1010$.

We do an inner induction on the number of irreducible components of x. If x is reducible, then u has denominator exponent $k - 1$ by Corollary 6.2.15 on page 122, and we can apply the outer induction hypothesis. Now suppose there is some j such that x_j is irreducible; then $x_j^\dagger x_j \neq 0000$ by Lemma 6.2.13 on page 121. Because of the evenness property noted above, there must exist some $\ell \neq j$ such that $x_j^\dagger x_j = x_\ell^\dagger x_\ell$. Applying Lemma 6.4.1 on page 146 to $u' = (u_j, u_\ell)^T$, we find a sequence $\vec{U}$ of row operations of types H and T, making $\rho_k(\vec{U}u')$ reducible. We can lift this to a two-level operation $\vec{U}_{[j,\ell]}$ acting on u; thus $\rho_k(\vec{U}_{[j,\ell]}u)$ has fewer irreducible components than $x = \rho_k(u)$, and the inner induction hypothesis applies. $\qquad\square$

Lemma 6.4.3 (Matrix decomposition)**.** *Let U be a unitary $n \times n$-matrix with entries in $\mathbb{D}[\omega]$. Then there exists a sequence $U_1, \ldots, U_h$ of one- and two-level unitary matrices of types X, H, T, and ω such that $U = U_1 \cdots U_h$.*

Proof. Equivalently, it suffices to show that there exist one- and two-level unitary matrices $V_1, \ldots, V_h$ of types X, H, T, and ω such that $V_h \cdots V_1 U = I$. This is an easy consequence of the column lemma, exactly as in e.g. [17, Sec. 4.5.1]. Specifically, first use the column lemma to find suitable one- and two-level row operations $V_1, \ldots, V_{h_1}$ such that the leftmost column of $V_{h_1} \cdots V_1 U$ is e_1. Because $V_{h_1} \cdots V_1 U$ is unitary, it is of the form

$$\left(\begin{array}{c|c} 1 & 0 \\ \hline 0 & U' \end{array} \right).$$

Now recursively find row operations to reduce U' to the identity matrix. $\square$

Example 6.4.4. We will decompose the matrix U from Example 6.2.18. We start with the first column u of U:

$$u = \frac{1}{\sqrt{2}^3}\begin{pmatrix} -\omega^3 + \omega - 1 \\ \omega^2 + \omega \\ \omega^3 + \omega^2 \\ -1 \end{pmatrix},$$

$$\rho_3(u) = \begin{pmatrix} 1011 \\ 0110 \\ 1100 \\ 0001 \end{pmatrix}, \quad \rho_3(u_j^\dagger u_j) = \begin{pmatrix} 0001 \\ 1010 \\ 1010 \\ 0001 \end{pmatrix}.$$

Rows 2 and 3 satisfy case 2 of Lemma 6.4.1 on page 146. As they are not aligned, first apply $T_{[2,3]}^3$ and then $H_{[2,3]}$. Rows 1 and 4 satisfy case 3. Applying $H_{[1,4]}T_{[1,4]}^2$, the residues become $\rho_3(u_1') = 0011$ and $\rho_3(u_4') = 1001$, which requires applying $H_{[1,4]}T_{[1,4]}$. We now have

$$H_{[1,4]}T_{[1,4]}H_{[1,4]}T_{[1,4]}^2 H_{[2,3]}T_{[2,3]}^3 u = v = \frac{1}{\sqrt{2}^2}\begin{pmatrix} 0 \\ 0 \\ \omega^2 + \omega \\ -\omega + 1 \end{pmatrix},$$

$$\rho_2(v) = \begin{pmatrix} 0000 \\ 0000 \\ 0110 \\ 0011 \end{pmatrix}, \quad \rho_2(v_j^\dagger v_j) = \begin{pmatrix} 0000 \\ 0000 \\ 1010 \\ 1010 \end{pmatrix}.$$

Rows 3 and 4 satisfy case 2, while rows 1 and 2 are already reduced. We reduce rows 3 and 4 by applying $H_{[3,4]}T_{[3,4]}$. Continuing, the first column is completely reduced to e_1 by further applying $\omega_{[1]}^7 X_{[1,4]}H_{[3,4]}T_{[3,4]}^3$. The complete decomposition of u is therefore given by

$$W_1 = \omega_{[1]}^7 X_{[1,4]}H_{[3,4]}T_{[3,4]}^3 H_{[3,4]}T_{[3,4]}$$

$$H_{[1,4]}T_{[1,4]}H_{[1,4]}T_{[1,4]}^2 H_{[2,3]}T_{[2,3]}^3.$$

Applying this to the original matrix U, we have $W_1 U =$

$$\frac{1}{\sqrt{2}^3} \begin{pmatrix} \sqrt{2}^3 & 0 & 0 & 0 \\ 0 & \omega^3-\omega^2+\omega+1 & -\omega^2-\omega-1 & \omega^2 \\ 0 & 0 & \omega^3+\omega^2-\omega+1 & \omega^3+\omega^2-\omega-1 \\ 0 & \omega^3+\omega^2+\omega+1 & \omega^2 & \omega^3-\omega^2+1 \end{pmatrix}.$$

Continuing with the rest of the columns, we find $W_2 = \omega_{[2]}^6 H_{[2,4]} T_{[2,4]}^3 H_{[2,4]} T_{[2,4]}$, $W_3 = \omega_{[3]}^4 H_{[3,4]} T_{[3,4]}^3 H_{[3,4]}$, and $W_4 = \omega_{[4]}^5$. We then have $U = W_1^\dagger W_2^\dagger W_3^\dagger W_4^\dagger$, or explicitly:

$$U = T_{[2,3]}^5 H_{[2,3]} T_{[1,4]}^6 H_{[1,4]} T_{[1,4]}^7 H_{[1,4]}$$

$$T_{[3,4]}^7 H_{[3,4]} T_{[3,4]}^5 H_{[3,4]} X_{[1,4]} \omega_{[1]}$$

$$T_{[2,4]}^7 H_{[2,4]} T_{[2,4]}^5 H_{[2,4]} \omega_{[2]}^2 H_{[3,4]} T_{[3,4]}^5 H_{[3,4]} \omega_{[3]}^4 \omega_{[4]}^3.$$

6.4.2 Main result

Consider the ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$, consisting of complex numbers of the form

$$\frac{1}{2^n}(a + bi + c\sqrt{2} + di\sqrt{2}),$$

where $n \in \mathbb{N}$ and $a, b, c, d \in \mathbb{Z}$. Our goal is to prove the following theorem, which was conjectured by Kliuchnikov et al. [12]:

Theorem 6.4.5. *Let U be a unitary $2^n \times 2^n$ matrix. Then the following are equivalent:*

> *(a) U can be exactly represented by a quantum circuit over the Clifford+T gate set, possibly using some finite number of ancillas that are initialized and finalized in state $|0\rangle$.*

> *(b) The entries of U belong to the ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$.*

Moreover, in (a), a single ancilla is always sufficient.

Proof. First note that, since all the elementary Clifford+T gates, as shown in (equation (6.27) on page 144), take their matrix entries in $\mathbb{D}[\omega] = \mathbb{Z}[\frac{1}{\sqrt{2}}, i]$, the implication (a) $\implies$ (b) is trivial. For the converse, let U be a unitary $2^n \times 2^n$ matrix with entries from $\mathbb{D}[\omega]$.

By Lemma 6.4.3, U can be decomposed into one- and two-level matrices of types X, H, T, and ω. It is well-known that each such matrix can be further decomposed into controlled-not gates and multiply-controlled X, H, T, and ω-gates, for example using Gray codes [17, Sec. 4.5.2]. But all of these gates have well-known exact representations in Clifford+T with ancillas, see e.g. [4, Fig. 4(a) and Fig. 9] (and noting that a controlled-ω gate is the same as a T-gate). This finishes the proof of (b) $\implies$ (a).

The final claim that needs to be proved is that a circuit for U can always be found using at most one ancilla. It is already known that for $n > 1$, an ancilla is sometimes necessary [12]. To show that a single ancilla is sufficient, in light of the above decomposition, it is enough to show that the following can be implemented with one ancilla:

    (a) a multiply-controlled X-gate;

    (b) a multiply-controlled H-gate;

    (c) a multiply-controlled T-gate.

We first recall from [4, Fig. 4(a)] that a singly-controlled Hadamard gate can be decomposed into Clifford+T gates with no ancillas:

We also recall that an n-fold controlled iX-gate can be represented using $O(n)$ Clifford+T gates with no ancillas. Namely, for $n = 1$, we have

and for $n \geq 2$, we can use

with further decompositions of the multiply-controlled not-gates as in [5, Lem. 7.2] and [17, Fig. 4.9]. We then obtain the following representations for (a)–(c), using only one ancilla:

$\square$
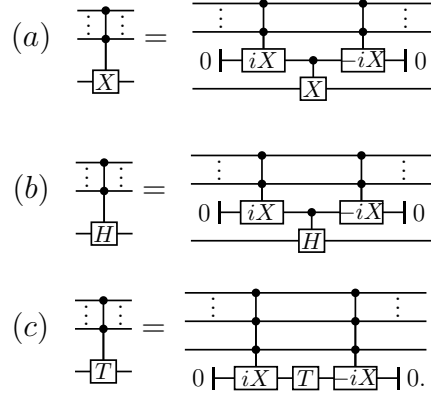
Remark 6.4.6. The fact that one ancilla is always sufficient in Theorem 6.4.5 on the next page is primarily of theoretical interest. In practice, one may assume that on most quantum computing architectures, ancillas are relatively cheap. Moreover, the use of additional ancillas can significantly reduce the size and depth of the generated circuits (see e.g. [24]).

6.4.3 The no-ancilla case

Lemma 6.4.7. *Under the hypotheses of Theorem 6.4.5, assume that* $\det U = 1$. *Then* U *can be exactly represented by a Clifford+T circuit with no ancillas.*

Proof. This requires only minor modifications to the proof of Theorem 6.4.5. First observe that whenever an operator of the form HT^m was used in the proof of Lemma 6.4.1 on page 146, we can instead use $T^{-m}(iH)T^m$ without altering the rest of the argument. In the base case of Lemma 6.4.2 on page 148, the operator $X_{[1,j]}$ can be replaced by $iX_{[1,j]}$. Also, in the base case of Lemma 6.4.2 on page 148, whenever $n \geq 2$, the operator $\omega_{[1]}$ can be replaced by $W_{[1,2]}$, where

$$W = \begin{pmatrix} \omega & 0 \\ 0 & \omega^{-1} \end{pmatrix}.$$

152

Therefore, the decomposition of Lemma 6.4.3 on page 149 can be performed so as to yield only two-level matrices of types

$$iX, \quad T^{-m}(iH)T^m, \quad \text{and } W, \tag{6.29}$$

plus at most one one-level matrix of type ω^m. But since all two-level matrices of types (equation (6.29)), as well as U itself, have determinant 1, it follows that $\omega^m = 1$. We finish the proof by observing that the multiply-controlled operators of types (equation (6.29)) possess ancilla-free Clifford+T representations, with the latter two given by

$\square$

As a corollary, we obtain a characterization of the n-qubit Clifford+T group (with no ancillas) for all n:

Corollary 6.4.8. *Let U be a unitary $2^n \times 2^n$ matrix. Then the following are equivalent:*

> *(a) U can be exactly represented by a quantum circuit over the Clifford+T gate set on n qubits with no ancillas.*

> *(b) The entries of U belong to the ring $\mathbb{Z}[\frac{1}{\sqrt{2}}, i]$, and:*

>> *• $\det U = 1$, if $n \geq 4$;*

>> *• $\det U \in \{-1, 1\}$, if $n = 3$;*

>> *• $\det U \in \{i, -1, -i, 1\}$, if $n = 2$;*

>> *• $\det U \in \{\omega, i, \omega^3, -1, \omega^5, -i, \omega^7, 1\}$, if $n \leq 1$.*

Proof. For (a) $\implies$ (b), it suffices to note that each of the generators of the Clifford+T group, regarded as an operation on n qubits, satisfies the conditions in (b). For (b) $\implies$

153

(a), let us define for convenience $d_0 = d_1 = \omega$, $d_2 = i$, $d_3 = -1$, and $d_n = 1$ for $n \geq 4$. First note that for all n, the Clifford+T group on n qubits (without ancillas) contains an element D_n whose determinant is d_n, namely $D_n = I$ for $n \geq 4$, $D_3 = T \otimes I \otimes I$, $D_2 = T \otimes I$, $D_1 = T$, and $D_0 = \omega$. Now consider some U satisfying (b). By assumption, $\det U = d_n^m$ for some m. Let $U' = U D_n^{-m}$, then $\det U' = 1$. By Lemma 6.4.7, U', and therefore U, is in the Clifford+T group with no ancillas. □

Remark 6.4.9. Note that the last condition in Corollary 6.4.8, namely that $\det U$ is a power of ω for $n \leq 1$, is of course redundant, as this already follows from $\det U \in \mathbb{Z}[\frac{1}{\sqrt{2}}, i]$ and $|\det U| = 1$. We stated the condition for consistency with the case $n \geq 2$.

Remark 6.4.10. The situation of Theorem 6.4.5 on page 151 and Corollary 6.4.8 on the previous page is analogous to the case of classical reversible circuits. It is well-known that the not-gate, controlled-not gate, and Toffoli gate generate all classical reversible functions on $n \leq 3$ bits. For $n \geq 4$ bits, they generate exactly those reversible boolean functions that define an *even permutation* of their inputs (or equivalently, those that have determinant 1 when viewed in matrix form) [16]; the addition of a single ancilla suffices to recover all boolean functions.

6.4.4 Complexity

The proof of Theorem 6.4.5 on page 151 immediately yields an algorithm, albeit not a very efficient one, for synthesizing a Clifford+T circuit with ancillas from a given operator U. We estimate the size of the generated circuits.

We first estimate the number of (one- and two-level) operations generated by the matrix decomposition of Lemma 6.4.3 on page 149. The row operation from Lemma 6.4.1 on page 146 requires only a constant number of operations. Reducing a single n-dimensional column from denominator exponent k to $k - 1$, as in the induction step of Lemma 6.4.2 on page 148, requires $O(n)$ operations; therefore, the number of operations required to reduce

the column completely is $O(nk)$.

Now consider applying Lemma 6.4.3 to an $n \times n$-matrix with least denominator exponent k. Reducing the first column requires $O(nk)$ operations, but unfortunately, it may *increase* the least denominator exponent of the rest of the matrix, in the worst case, to $3k$. Namely, each row operation of Lemma 6.4.1 potentially increases the denominator exponent by 2, and any given row may be subject to up to k row operations, resulting in a worst-case increase of its denominator exponent from k to $3k$ during the reduction of the first column. It follows that reducing the second column requires up to $O(3(n-1)k)$ operations, reducing the third column requires up to $O(9(n-2)k)$ operations, and so on. Using the identity $\sum_{j=0}^{n-1} 3^j(n-j) = (3^{n+1} - 2n - 3)/4$, this results in a total of $O(3^n k)$ one- and two-level operations for Lemma 6.4.3.

In the context of Theorem 6.4.5 on page 151, we are dealing with n qubits, i.e., a $2^n \times 2^n$-operator, which therefore decomposes into $O(3^{2^n} k)$ two-level operations. Using one ancilla, each two-level operation can be decomposed into $O(n)$ Clifford+T gates, resulting in a total gate count of $O(3^{2^n} nk)$ elementary Clifford+T gates.

Chapter 7

Conclusions and future work

Bibliography

[1] S. Abramsky. A structural approach to reversible computation. *Theoretical Computer Science*, 347(3):441–464, 2005.

[2] S. Abramsky and B. Coecke. Physical traces: Quantum vs. classical information processing. *Electr. Notes Theor. Comput. Sci*, 69, 2002.

[3] S. Abramsky and B. Coecke. Abstract physical traces. *Theory and Applications of Categories*, 14:114–124, 2005.

[4] Matthew Amy, Dmitri Maslov, Michele Mosca, and Martin Roetteler. A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. Version 2, arXiv:1206.0758v2, August 2012.

[5] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52:3457–3467, 1995. Available from arXiv:quant-ph/9503016v1.

[6] Erik Barendsen, Inge Bethke, Jan Heering, Richard Kennaway, Paul Klint, Vincent van Oostrom, Femke van Raamsdonk, Fer-Jan de Vries, and Hans Zantema. Cambridge University Press, The Edinburgh Building, Cambridge, CB2 2RU, UK, 2003.

[7] Charles H. Bennet. Logical reversibility of computation. *IBM Journal of Research and Development*, 6:525–532, 1973.

[8] Alex Bocharov and Krysta M. Svore. Resource-optimal single-qubit quantum circuits. *Physical Review Letters*, 109:190501 (5 pages), 2012. Also available from arXiv:1206.3223.

[9] J.R.B. Cockett, Xiuzhan Guo, and Pieter Hofstra. Range categories ii: Towards regularity. Submitted for Publication, June 2012.

[10] David Deutsch. Quantum computational networks. *Proceedings of the Royal Society of London Ser. A*, A425:73–90, 1989.

[11] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation.* Pearson/Addison Wesley, 3rd edition, 2007.

[12] Vadym Kliuchnikov, Dmitri Maslov, and Michele Mosca. Fast and efficient exact synthesis of single qubit unitaries generated by Clifford and T gates. arXiv:1206.5236v2, June 2012.

[13] Joachim Kock. *Frobenius Algebras and 2D Topological Quantum Field Theories.* Number 59 in London Mathematical Society Student Texts. Cambridge University Press, 2004.

[14] Saunders Mac Lane. *Categories for the Working Mathematician.* Springer Verlag, Berlin, Heidelberg, Germany, second edition, 1997. ISBN 0-387-98403-8. Dewey QA169.M33 1998.

[15] Ken Matsumoto and Kazuyuki Amano. Representation of quantum circuits with clifford and $\pi/8$ gates. arXiv:0806.3834v1, June 2008.

[16] Julien Musset. Générateurs et relations pour les circuits booléens réversibles. Technical Report 97-32, Institut de Mathématiques de Luminy, 1997. Available from http://iml.univ-mrs.fr/editions/.

[17] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press, The Edinburgh Building, Cambridge CB2 2RU, UK, 2000. ISBN 0 521 63235 8.

[18] Robin Cockett. Category theory for computer science. Available at http://pages.cpsc.ucalgary.ca/~robin/class/617/notes.pdf, October 2009.

[19] Robin Cockett and Stephen Lack. Restriction categories I: categories of partial maps. *Theoretical Computer Science*, 270:223–259, 2002.

[20] Robin Cockett and Stephen Lack. Restriction categories II: Partial map classification. *Theoretical Computer Science*, 294:61–102, 2003.

[21] Robin Cockett and Stephen Lack. Restriction categories III: colimits, partial limits, and extensivity. *Mathematical Structures in Computer Science*, 17(4):775–817, 2007. Available at http://au.arxiv.org/abs/math/0610500v1.

[22] Peter Selinger. Towards a quantum programming language. *Mathematical Structures in Computer Science*, 14(4):527–586, 2004.

[23] Peter Selinger. Dagger compact closed categories and completely positive maps. In *Proceedings of the 3rd International Workshop on Quantum Programming Languages, Chicago*, 2005.

[24] Peter Selinger. Quantum circuits of T-depth one. *Physical Review A*, 2013. To appear. Available from arXiv:1210.0974.

[25] Vlatko Vedral, Adriano Barenco, and Artur Ekert. Quantum networks for elementary arithmetic operations. *Physical Review A*, 54:147, 1995.