

# Quantum circuit oracles for Abstract Machine computations

Peter Hines<sup>1</sup>

*Dept. of Computer Science, University of York, York, U.K.*

---

## Abstract

This paper considers a very general model of computation via conditional iteration, the *abstract machines* of [23], and studies the conditions under which these describe reversible computations. Using this, we demonstrate how to construct quantum circuits that act as oracles for these Abstract Machines.

For a classical computation with worst-case performance  $T$ , the resulting quantum circuit requires an ancilla of  $1 + \log(T)$  qubits, and takes  $O(T)$  steps. The ancilla starts and finishes in the constant state  $|0\rangle$ , so garbage collection is performed automatically.

*Key words:* Quantum Computing, Circuit model, Conditional Iteration, Abstract Machines.

---

## 1 Introduction

Most, if not all, quantum algorithms rely on an oracle. Informally, an oracle for a classical reversible computation  $f$  is a unitary map that acts as  $f$  on some subset of the computational basis, and acts linearly on arbitrary superpositions of these basis states. Oracles often also rely on ancillary quantum registers — these are required to start and finish in some fixed computational basis state, disentangled from either the input or the output of the computation.

In creating an oracle, it is standard to assume that the classical computation is determined by a reversible acyclic Boolean circuit [41]. Although a large standard toolkit exists for creating reversible circuits from irreversible acyclic Boolean circuits, and translating them into quantum oracles [7,8], in practice creating an oracle is often a bottleneck in quantum computations [37]. This

---

<sup>1</sup> Research supported by EU project QICS (033763)

is due to the extraordinarily limited forms of classical computation that may be translated into quantum circuits: arbitrary classical computations must be rewritten in entirely reversible form, without any notion of feedback, conditional iteration, or similar. Further, any auxillary workspace used in the computation must start and end in some constant fixed state.

The purpose of this paper is to remove at least one of these obstacles to creating quantum oracles, by demonstrating how to create oracles, given explicitly as quantum circuits, from classical computations based on *conditional iteration* or *conditional halting*.

We demonstrate a systematic method of producing quantum circuits that compute the same function as a classical iterative computation on the computational basis, and are superposition-preserving on arbitrary inputs – i.e. we give a general prescription for constructing oracles based on conditional iteration.

The notion of ‘computation via conditional iteration’ is formalised using the *Abstract Machines* paradigm of [23] (although prior knowledge of [23] is not a prerequisite for this paper). An abstract machine is simply a set, containing distinguished *start* and *halt* subsets, along with an evolution operator. Computation takes place by starting with a member of the start subset, and iterating the evolution operator until the halting subset is reached. Although this is a simple, almost trivial, definition, many computational devices (e.g. Turing machines, von Neumann architectures, cellular automata, etc.) may be expressed in this form. Abstract machines give rise to surprisingly rich structures, allowing for (for example) domain-theoretic and categorical analyses of such computational paradigms – see [23] for details.

### 1.1 Historical context

Quantum computing and conditional iteration have a long and uneasy history. Although D. Deutsch’s original conception of a ‘quantum Turing machine’ [12] defined computation in terms of conditional halting, his approach was shown to be fundamentally flawed by J. Myers [40]. N. Linden and S. Popescu phrased the problem more generally, as a question about general quantum algorithms [34], and demonstrated how using conditional iteration within quantum algorithms always seemed to lead to uncontrollable entanglement with an auxillary space (thus forcing decoherence, and ruling out the use of conditional iteration in producing oracles). Linden and Popescu posed the (still open) question of which computations based on conditional iteration can or cannot be restructured to avoid such decoherence.

These results serve to emphasise that reversibility alone is not sufficient to

produce a quantum analogue of a classical computation — although the dynamics of a quantum computational system may be entirely unitary, attempts to define computations in terms of conditional halting seem to cause decoherence (via entanglement with an ancilla). Thus, although there is a general toolkit for modelling irreversible Turing machine computations by reversible Turing machines [33,31,8,32], this translation alone is not enough to produce quantum oracles from Turing machine computations.

An alternative approach to quantum Turing machines was taken in [9], where *unconditional* iteration (i.e. halting after some fixed number of steps) is used. Such ‘properly halting quantum Turing machines’ are equivalent to standard quantum circuits — however, the problem of taking a (classical, irreversible) Turing machine computation, and re-writing it so that it is not only *reversible*, but terminates after a fixed number of steps *regardless of the input*, is trivially equivalent to the problem laid out in [34]. For arbitrary Turing machines, this is of course undecidable in general, due to the undecidability of halting: see [29,38,39] for various no-go theorems<sup>2</sup>.

Finally, conditional iteration in quantum computation is often based on *classical conditionals*, conditioned on measurements of a quantum system (as in, for example, [42]). Although this leads to many interesting phenomena, it also, of course, also leads to decoherence, and is not suitable for translating a classical computation based on conditional iteration into a quantum oracle.

## 2 Abstract Machines and classical iteration

Our description of conditional iteration is based on the very general ‘abstract machines’ of [21,23,22]. We do not present the full theory here — rather we study a special case where both *finiteness* and *reversibility* are assumed.

**Definition 1** *An (finite reversible) Abstract Machine, or frAM  $\mathcal{M}$  consists of:*

- *A finite configuration set  $Y$ .*
- *A bijection  $\mathcal{P} : Y \rightarrow Y$  called the primitive evolution.*

---

<sup>2</sup> For readers familiar with [23], space-bounded Turing machines were analysed in great detail, as canonical examples of Abstract Machines. An application of this current paper is indeed a procedure for creating oracles for space-bounded Turing machine computations in the quantum circuit paradigm. We emphasise that this does not contradict the above no-go theorems — the essential difference being that such undecidability results are not applicable in the space-bounded case. However, the details of this construction are sufficiently non-trivial to make this the subject of a paper in its own right [26].

- A subset  $S$  of the configuration set called the **start-halt subset**.

We refer to the complement of the start-halt subset  $W = Y \setminus S$  as the **working subset**.

A **computation** of a frAM proceeds as follows:

- (1) The input is some starting configuration  $s \in S \subseteq Y$ .
- (2) The primitive evolution is applied:  $s \mapsto \mathcal{P}(s)$ .
- (3) Step (2) is repeated until some halting configuration  $t \in S$  is reached – this is then the output.

For a frAM  $\mathcal{M}$ , this iterative procedure defines a function  $\{\mathcal{M}\} : S \rightarrow S$  that we call the **function computed by  $\mathcal{M}$** . In Corollary 10 we will show that  $\{\mathcal{M}\} : S \rightarrow S$  is in fact a bijection.

By definition  $t = \{\mathcal{M}\}(s)$  implies that  $t = \mathcal{P}^k(s)$ , for some  $k \geq 1$ . We emphasise that this (non-zero, possibly non-unique) integer is not a constant – rather, it is determined by the input  $s$ .

For such ideas from a different perspective, we observe the similarity between this scheme for computation, and the ‘reversible iteration’ primitive of the reversible programming language JANUS [35].

In Definition 1 above, a single subset acts as both a starting and halting subset. This is non-standard in automata theory and related fields of theoretical computer science. However, it is not a major restriction — a more general case with distinct (but equal sized) starting and halting subsets  $S, H \subseteq Y$  may be simulated by modifying the primitive evolution  $\mathcal{P}$  using some bijection  $\sigma : Y \rightarrow Y$  that interchanges  $S$  and  $H$ , and is the identity elsewhere. However, the algebra associated with having a single start/halt subset is significantly more elegant, so without loss of generality we will work with abstract machines of this form.

## 2.1 Conventions of frAMs

In order to allow for an easy translation into the quantum circuit paradigm, we assume that our frAMs are of a certain special form, as described below. We emphasise that these assumptions are made *without loss of generality*. They are either labelling conventions, or can be imposed by introducing ‘padding’ that does not interact with the computation in any way. In neither case do they impose any restrictions on the classical frAMs for which we may create

quantum oracles.

### 2.1.1 Conventions for finite reversible Abstract Machines

- (1) The configuration set is  $Y = \{0, \dots, 2^n - 1\}$  for some integer  $n$ .
- (2) The start-halt subset is exactly those configurations whose most significant bit, or **start-halt bit**, is 0. Thus  $S = \{0, \dots, 2^{n-1} - 1\}$  and  $W = \{2^{n-1}, \dots, 2^n - 1\}$ .

In the scheme we present, upper bounds to the number of steps before termination will be important. An upper bound (not necessarily a *least* upper bound) for number of steps before termination is simply an integer  $T > 0$  such that  $\{\mathcal{M}\}(p) = q$  implies  $q = \mathcal{P}^K(p)$ , for some  $K < T$ .

Although  $T = 2^{n-1} + 1$  is an obvious such upper bound, in general this will be a vast overestimate. Sometimes a much tighter bound may be deduced combinatorially, or by detailed analysis of the Abstract Machine. However, given a quantum computer, a suitable (although not in general optimal) bound may be found using standard techniques:

**Theorem 2** *Given a frAM  $\mathcal{M} = (Y, \mathcal{P}, S)$ , an upper bound to the number of steps before termination may be found by quantum period-finding.*

**PROOF.** There are numerous subtleties associated with quantum period-finding, so a discussion of this is postponed to Appendix III.  $\square$

### 2.2 An example frAM computation

An example of a finite reversible abstract machine  $\mathcal{M} = (Y, \mathcal{P}, S)$  satisfying the conventions of Section 2.1 is given in Figure 1. Here, the configuration set is all 4 bit binary words,

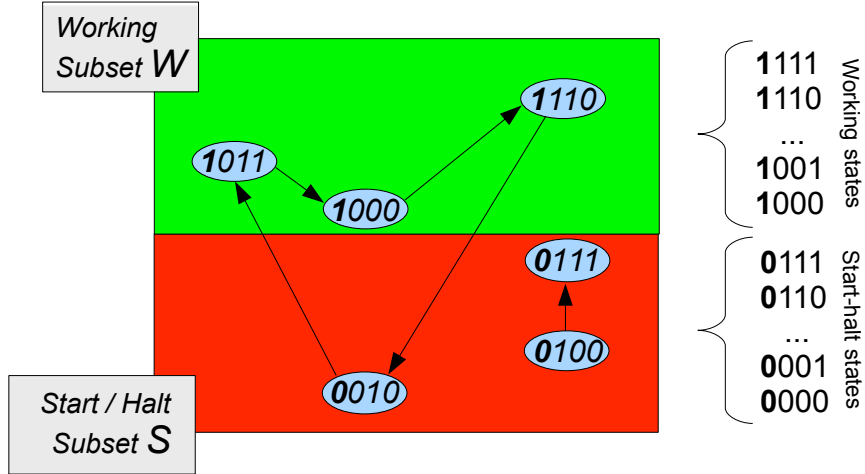
$$Y = \{0000, 0001, \dots, 1110, 1111\}$$

and the primitive evolution  $\mathcal{P}$  is given by,

*“Flip all bits in a word, then apply a cyclic left-shift”.*

As specified above, the most significant, or start-halt, bit determines the

Fig. 1. An example frAM computation



start/halt subset, so

$$S = \{0000, \dots, 0111\} \text{ and } W = \{1000, \dots, 1111\}$$

Two distinct computations, from the start-halt subspace to itself, are shown in Figure 1.

Simple combinatorics shows that the number of steps before termination can be no greater than 5, and a slightly more in-depth analysis shows that this figure (i.e. 5 steps) is in fact an over-estimate.

### 3 The problem solved by this paper

Following the above definitions and examples, we are now able to state precisely the problem solved by this paper:

*“Given a quantum oracle  $U_{\mathcal{P}}$  for the primitive evolution of a (finite, reversible) Abstract Machine, how may we give a quantum oracle  $U_{\mathcal{M}}$  for the function computed by this abstract machine?”*

Disregarding complexity considerations, a solution to the above problem is immediate. We simply perform the computational basis computation on every possible input, create a lookup table, and use any of the standard techniques to create a quantum circuit that implements this function. The complexity of taking this approach is also, of course, ridiculous.

The solution presented in this paper takes  $O(T)$  calls to the oracle  $U_{\mathcal{P}}$ , and requires an ancilla of  $1 + \log(T)$  qubits, for a computation with worst-case running time bounded by  $T$  steps. Note that, in contrast to the exhaustive search above, complexity is determined by the maximal number of steps to termination, rather than the size of the configuration set  $Y$ .

## 4 The classical theory

We now give an exposition of the classical theory of abstract machines, as developed in [21,23,22], and the required algebra. As this paper is intended to be an application of the general theory, we present the minimal subset required to reach our final result. We then give a standard embedding of this classical theory into the algebra appropriate to quantum circuits, and use this to prove correctness of the procedure laid out for constructing quantum oracles.

## 5 Algebraic preliminaries

The required algebra is based on the theory of partial bijections (i.e. partial functions that are bijective where defined) — we refer to [30] for comprehensive background.

**Definition 3** *Given sets  $A, B$ , a **partial bijection**  $f : A \rightarrow B$  is a subset of  $B \times A$  satisfying  $a = a' \Leftrightarrow b = b' \quad \forall (b, a), (b', a') \in f$ . Given  $(b, a) \in f$ , we commonly use functional notation, and write  $b = f(a)$ .*

*Given a partial bijection  $f : A \rightarrow B$ , then the **domain** and **image** of  $f$  are respectively the subsets  $\text{dom}(f) \subseteq A$  and  $\text{im}(f) \subseteq B$  defined by*

- $\text{dom}(f) = \{a \in A : \exists b \in B \text{ s.t. } (b, a) \in f\}$
- $\text{im}(f) = \{b \in B : \exists a \in A \text{ s.t. } (b, a) \in f\}$

*The **composite** of partial bijections  $f : A \rightarrow B$  and  $g : B \rightarrow C$  is the partial bijection  $gf$  defined by*

$$(c, a) \in gf \Leftrightarrow \exists b \in B : (c, b) \in g \text{ and } (b, a) \in f$$

*In functional notation,  $b = f(a)$  and  $c = g(b) \Rightarrow c = gf(a)$ .*

*Given a partial bijection  $f : A \rightarrow B$ , its **generalised inverse** is the partial bijection  $f^{-1} : B \rightarrow A$  defined by*

$$(a, b) \in f^{-1} \Leftrightarrow (b, a) \in f$$

In functional notation,

$$f(a) = b \Leftrightarrow f^{-1}(b) = a$$

Note that generalised inverse do not, in general, satisfy  $f^{-1}f = 1_A$ . Rather,  $f^{-1}f$  is a partial identity on  $A$  (i.e. a partial bijection  $e : A \rightarrow A$  satisfying  $(a', a) \in e \Rightarrow a = a'$ ).

Of particular importance in this paper are the notions of summation and matrix representations of partial bijections.

**Definition 4** An indexed family of partial bijections  $\{f_i\}_{i \in I}$  is **summable** exactly when, for all  $i \neq j \in I$ , the partial bijections  $f_i$  and  $f_j$  have disjoint domains and images. Given a summable indexed family  $\{f_i : A \rightarrow B\}_{i \in I}$ , the **sum** of this family is exactly the set-theoretic union  $\sum_{i \in I} f_i \stackrel{\text{def.}}{=} \bigcup_{i \in I} f_i$ . We refer to [15] for a study of the properties of this summation. We also use the notation  $(f_1 + \dots + f_n) : A \rightarrow B$  for indexed families of finite numbers of elements. For two-element families,  $f_1 + f_2$  is the partial bijection with domain  $\text{dom}(f_1) \cup \text{dom}(f_2)$  specified by

$$(f_1 + f_2)(a) = \begin{cases} f_1(a) & a \in \text{dom}(f_1), \\ f_2(a) & a \in \text{dom}(f_2). \end{cases}$$

The above notion of partial summation of partial bijections allows us to use matrix formalism for partial bijections, as follows:

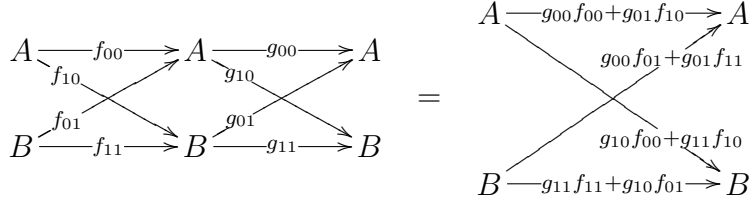
**Theorem 5** Consider a set  $Y$ , decomposed into the disjoint union of two subsets  $Y = A \uplus B$ . A partial bijection  $f : Y \rightarrow Y$  may be given a **matrix representation**, as

$$f = \begin{pmatrix} f_{00} & f_{01} \\ f_{10} & f_{11} \end{pmatrix} \quad \text{where} \quad \begin{array}{ll} f_{00} : A \rightarrow A & f_{01} : B \rightarrow A \\ f_{10} : A \rightarrow B & f_{11} : B \rightarrow B \end{array}$$

By interpreting multiplication as the composition of partial bijections, and summation as the addition of Definition 4, the matrix representation of  $gf : A \uplus B \rightarrow A \uplus B$  is given in terms of the matrix representations of  $g, f : A \uplus B \rightarrow A \uplus B$ , by the usual formula:

$$\begin{pmatrix} g_{00} & g_{01} \\ g_{10} & g_{11} \end{pmatrix} \begin{pmatrix} f_{00} & f_{01} \\ f_{10} & f_{11} \end{pmatrix} = \begin{pmatrix} g_{00}f_{00} + g_{01}f_{10} & g_{00}f_{01} + g_{01}f_{11} \\ g_{10}f_{00} + g_{11}f_{10} & g_{10}f_{01} + g_{11}f_{11} \end{pmatrix}$$

Fig. 2. Matrix composition as ‘summing over paths’



**PROOF.** We refer to [20] for this, and [15] for a general category-theoretic setting. Alternative proofs and significant applications are given in [5,16].  $\square$

The above formula for matrix composition has a simple diagrammatic interpretation as ‘summing over labelled paths’, as shown in figure 2.

### 5.1 The ‘resolution’ of a partial bijection

We now give another operation on matrices of partial bijections, with a similar diagrammatic interpretation to Figure 2. This will allow us to give an explicit description of the function computed by a (finite, reversible) abstract machine.

**Definition 6** Let  $f : Y \rightarrow Y$  be a partial bijection, and let  $Y$  be given as the disjoint union of two subsets, as  $Y = S \uplus W$ . By Theorem 5 we may give a matrix representation for  $f$ , as

$$f = \begin{pmatrix} f_{00} & f_{01} \\ f_{10} & f_{11} \end{pmatrix} : S \uplus W \rightarrow S \uplus W$$

The **resolution** of  $f$  at  $S$  is the partial bijection defined in terms of this matrix representation, as

$$Res_S(f) = f_{00} + \sum_{j=0}^{\infty} f_{01} f_{11}^j f_{10}$$

We refer to Theorem 7 below for a proof that this is indeed a partial bijection.

Diagrammatically, as shown in Figure 3, this is again simply taking the sum over paths — albeit this time with the introduction of a partial feedback loop.

Fig. 3. Resolution as ‘summing over paths’

$$\begin{array}{ccc}
 \begin{array}{ccc}
 S & \xrightarrow{f_{00}} & S \\
 & \searrow f_{10} & \nearrow \\
 & & W \\
 W & \xrightarrow{f_{11}} & W \\
 & \nwarrow f_{01} & \nearrow \\
 & & S
 \end{array} & = & \begin{array}{ccc}
 S & \xrightarrow{f_{00}} & S \\
 & \searrow f_{10} & \nearrow \\
 & & W \\
 & & \circlearrowleft f_{11} \\
 & & \nearrow \\
 & & S
 \end{array} & = & S \xrightarrow{f_{00} + \sum_{j=0}^{\infty} f_{01}(f_{11})^j f_{01}} S
 \end{array}$$

### 5.2 A general category-theoretic setting for Resolution

Readers familiar with the appropriate category theory will of course recognise the ‘Resolution’ given above as a special case of a category-theoretic trace [27,2]. The exact formula given in Definition 6 above appears (in the context of the category of relations on sets, and the disjoint union) in [27,2], using the notation  $Tr_{X,Y}^U(R)$ . We do not use this notation or terminology, to avoid the inevitable confusion with the usual familiar Trace on linear operators (indeed, this is also an example of a categorical trace [3]). Instead, we use the alternative term ‘Resolution’ following [11,14,23].

There is no space here to give a summary of the history and development of categorical traces in general. We refer to [27,2] for the origins of the theory, [19,18] for a good overview, and in the remainder of the paper simply concentrate on the special case of categorical traces of partial bijections on sets.

### 5.3 Resolution and reversibility

As mentioned above, the formula of Definition 6 appears in [27,2] in the context of Relations and Partial functions on sets. In [20,15,5] it is demonstrated that the category of partial bijections also has such a categorical trace. Using the notation of this paper, this implies the following result:

**Theorem 7** *Let*

$$f = \begin{pmatrix} f_{00} & f_{01} \\ f_{10} & f_{11} \end{pmatrix} : S \uplus W \rightarrow S \uplus W$$

*be the matrix representation of a partial bijection. Then the family  $\{f_{00}\} \cup$*

$\{f_{01}f_{11}^j f_{10}\}_{j=0}^{\infty}$  is a summable family of partial bijections, and thus

$$\text{Res}_S(f) = f_{00} + \sum_{j=0}^{\infty} f_{01}f_{11}^j f_{10}$$

is a partial bijection.

**PROOF.** We refer to [20,15,5] for a range of proofs of this result, in a more general category-theoretic setting.  $\square$

This result is significantly strengthened in [4], where the following result is proved:

**Theorem 8** *Let  $f : S \uplus W \rightarrow S \uplus W$  be a bijection (i.e. isomorphism) between finite sets. Then  $\text{Res}_S(f) : S \rightarrow S$  is itself a bijection.*

**PROOF.** This is explicitly proved in [4] and (according to the author of [4]) significantly prefigured in [28].  $\square$

## 6 The function computed by a frAM

The above algebraic and categorical background allows us to describe the partial function computed by an abstract machine in terms of its primitive evolution, as follows:

**Theorem 9** *Let  $\mathcal{R} = (Y, \mathcal{P}, S)$  be a finite reversible abstract machine, with  $\mathcal{P} : Y \rightarrow Y$  given in matrix form as*

$$\mathcal{P} = \begin{pmatrix} \mathcal{P}_{00} & \mathcal{P}_{01} \\ \mathcal{P}_{10} & \mathcal{P}_{11} \end{pmatrix} : S \uplus W \rightarrow S \uplus W$$

*Then  $\text{Res}_S(\mathcal{P}) : S \rightarrow S$ , as defined in Definition 6, is exactly the function  $\{\mathcal{M}\} : S \rightarrow S$  computed by the iterative procedure of Definition 1.*

**PROOF.** Although intuitively obvious, this is surprisingly difficult to prove. A proof is given in [23] (special case of Theorem 27), using domain-theoretic methods.  $\square$

**Corollary 10** *Let  $M = (Y, \mathcal{P}, S)$  be a frAM. Then the function  $\{M\} : S \rightarrow S$  computed by this machine is a bijection.*

**PROOF.** This is immediate from Theorems 8, 9 above.  $\square$

## 7 Embedding partial bijections into linear maps

We now consider how partial bijections between sets may be ‘lifted’ to linear maps between complex Hilbert spaces, via their action on some fixed orthonormal basis. This is a special case of a general construction presented in [6].

So far, in order to make this paper accessible for a general audience, we have avoided explicit category theory. However, it would be perverse to present this section in any other way. For a brief non-rigorous summary, we refer to the table in Figure 4., and for a full categorical treatment, we refer to [6].

**Definition 11** *The category of finite partial bijections, denoted  $\mathbf{fpBij}$  has as objects the proper class of all finite sets, and as arrows, partial bijections between these sets. Composition of arrows is given by the usual formula for composition of partial bijections given in Definition 3.*

*The category of finite-dimensional Hilbert spaces, denoted  $\mathbf{Hilb}_{\mathbf{FD}}$  has as objects the proper class of all finite-dimensional complex Hilbert spaces, and as arrows linear maps between these spaces. Composition of linear maps is given by the usual definition.*

We now introduce an injective functor from  $\mathbf{fpBij}$  to  $\mathbf{Hilb}_{\mathbf{FD}}$ . This is based on the  $l_2$  functor of [6] — however, we simplify the situation significantly by giving a *basis-dependent, covariant* version of this functor.

**Definition 12** *The functor  $l_2 : \mathbf{fpBij} \rightarrow \mathbf{Hilb}_{\mathbf{FD}}$  is defined by:*

- **(On Objects)** *Given a set  $X \in \mathit{Ob}(\mathbf{fpBij})$ , then the Hilbert space  $l_2(X)$  is defined to be the  $|X|$ -dimensional space with orthonormal basis  $\{|x\rangle\}_{x \in X}$ .*
- **(On Arrows)** *Given a partial bijection  $f \in \mathbf{fpBij}(X, Y)$ , the linear map  $l_2(f) \in \mathbf{Hilb}_{\mathbf{FD}}(l_2(X), l_2(Y))$  is defined in terms of basis elements by*

$$l_2(f) |x\rangle = \begin{cases} |f(x)\rangle & x \in \mathit{dom}(f) \\ 0 & \text{otherwise.} \end{cases}$$

*It is immediate that  $l_2 : \mathbf{fpBij} \rightarrow \mathbf{Hilb}_{\mathbf{FD}}$  is indeed a covariant functor.*

The connection between the functor  $l_2 : \mathbf{fpBij} \rightarrow \mathbf{Hilb}_{\mathbf{FD}}$  and the concept of an oracle for a classical reversible function (as in Section 1) is hopefully immediate: Given a bijection  $f$ , then the unitary  $l_2(f)$  is an oracle for  $f$ .

Readers familiar with [6] will observe that we have used dualities inherent in the category of Hilbert spaces to produce a covariant version of what is usually a contravariant functor. In doing this, we have been forced to specify distinguished bases for the Hilbert spaces considered. From many points of view, basis-independence is to be preferred. However, for the concrete application presented in this paper, we are working within the circuit model of quantum computation, where a fixed orthonormal basis (i.e. the *computational basis*) is assumed. Thus, the version of  $l_2$  presented above matches the usual intuition of ‘creating a quantum oracle’.

The following facts about this functor will be useful:

**Theorem 13**

- (1)  $l_2$  is injective, on both objects and arrows.
- (2) For all  $f \in \mathbf{fpBij}(X, Y)$ ,  $l_2(f)$  is a partial isometry.
- (3) When  $f \in \mathbf{fpBij}(X, Y)$  is an isomorphism, then  $l_2(f)$  is a unitary map.
- (4) When  $e, e' \in \mathbf{fpBij}(X, X)$  are partial identities, then  $l_2(e)$  and  $l_2(e')$  are commuting projectors.
- (5) The functor  $l_2 : \mathbf{fpBij} \rightarrow \mathbf{Hilb}_{\mathbf{FD}}$  maps the generalised inverse to the adjoint: that is,  $l_2(f^{-1}) = (l_2(f))^\dagger$ , for all  $f \in \mathbf{fpBij}(X, Y)$ .
- (6) Given a summable family of partial bijections  $\{f_i \in \mathbf{fpBij}(X, Y)\}_{i \in I}$ , then

$$l_2\left(\sum_{i \in I} f_i\right) = \sum_{i \in I} l_2(f_i)$$

where the sum on the l.h.s. is as given in Definition 4, and the sum on the r.h.s. is the usual component-wise summation of linear maps.

**PROOF.** We refer to either [6] or [16] for proofs.  $\square$

As well as being a covariant functor,  $l_2 : \mathbf{fpBij} \rightarrow \mathbf{Hilb}_{\mathbf{FD}}$  is a monoidal functor, with respect to two distinct monoidal structures on  $\mathbf{fpBij}$ .

**Theorem 14** *The category  $\mathbf{fpBij}$  is a symmetric monoidal category, with respect to both the disjoint union  $\uplus$  and the Cartesian product  $\times$ . Further, the  $l_2 : \mathbf{fpBij} \rightarrow \mathbf{Hilb}_{\mathbf{FD}}$  functor preserves both these monoidal structures, mapping  $\uplus$  to the **direct sum**  $\oplus$ , and mapping  $\times$  to the **tensor product**  $\otimes$ .*

**PROOF.** These are standard results of either [6] or [16].  $\square$

Fig. 4. Embedding partial bijections into linear maps  
*The Action of the functor  $l_2 : \mathbf{fpBij} \rightarrow \mathbf{Hilb}_{\mathbf{FD}}$*

$\mathbf{fpBij}$	$\mathbf{Hilb}_{\mathbf{FD}}$
Set $X = \{x_1 \dots x_n\}$	Hilbert space $l_2(X)$ , with orthonormal basis $\{ x_1\rangle, \dots,  x_n\rangle\}$
Partial Bijection $f(x_i) = y_j$	Partial Isometry $l_2(f)  x_i\rangle =  y_j\rangle$
Bijection	Unitary Map
Partial identities	(Commuting) projectors
Generalised inverse $( )^{-1}$	Adjoint $( )^\dagger$
Summation of partial bijections	Component-wise sum of linear maps
Disjoint union $\uplus$	Direct sum $\oplus$
Cartesian product $\times$	Tensor product $\otimes$

The above results may be summarised in the table of Figure 4., a slightly modified form of which is presented in [16].

## 8 The direct sum in the circuit paradigm

As noted in Section 5.2, the Resolution — a key formula in understanding abstract machine computations — is a categorical trace on  $\mathbf{fpBij}$ . However, it is a trace on the disjoint union, rather than the Cartesian product.

The functor  $l_2 : \mathbf{fpBij} \rightarrow \mathbf{Hilb}_{\mathbf{FD}}$ , which can reasonably be considered as formalising the notion of an oracle, maps the disjoint union  $\uplus$  to the direct sum  $\oplus$ . We therefore present the standard interpretation of the direct sum in the circuit paradigm, in terms of conditional operations.

**Definition 15** *Let  $U, V$  be unitary operations on  $n$  qubits. The **controlled operations**  $Ctrl_0U$  and  $Ctrl_1V$ , called **control-on-0** and **control-on-1** respectively, are the  $n + 1$  qubit operations defined by:*

$$Ctrl_0U |0\rangle |\psi\rangle = |0\rangle U |\psi\rangle \quad \text{and} \quad Ctrl_0U |1\rangle |\psi\rangle = |1\rangle |\psi\rangle$$

$$Ctrl_1V |0\rangle |\psi\rangle = |0\rangle |\psi\rangle \quad \text{and} \quad Ctrl_1V |1\rangle |\psi\rangle = |1\rangle V |\psi\rangle$$

These have the standard circuit representations shown in figure 5. When we

Fig. 5. Quantum circuits for ‘Control on 0’ and ‘Control on 1’

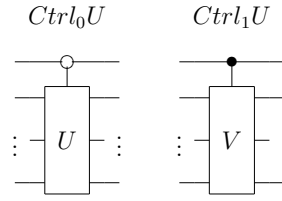
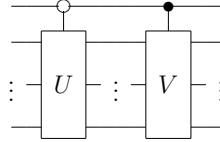


Fig. 6. A circuit for the direct sum  $U \oplus V$



denote the  $n$ -qubit identity operation by  $I_n$ , it is immediate that these operations have the following matrix representations:

$$C_0U = \begin{pmatrix} U & \mathbf{0} \\ \mathbf{0} & I_n \end{pmatrix}, \quad C_1V = \begin{pmatrix} I_n & \mathbf{0} \\ \mathbf{0} & V \end{pmatrix}$$

**Theorem 16** *Let  $U, V$  be  $n$ -qubit unitary operations. Then the direct sum*

$$U \oplus V = \begin{pmatrix} U & \mathbf{0} \\ \mathbf{0} & V \end{pmatrix}$$

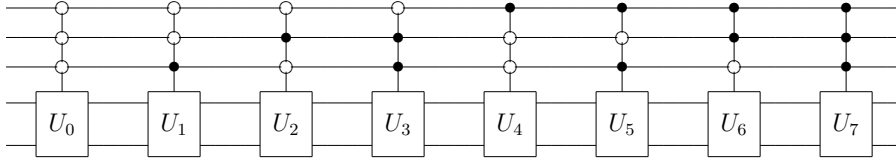
*is the  $n + 1$  qubit operation given by the composite*

$$U \oplus V = Ctrl_0U.Ctrl_1V = Ctrl_1V.Ctrl_0U$$

**PROOF.** This is immediate from comparing the matrix representations of the controlled operations with the matrix representation of the direct sum. A quantum circuit for the direct sum is shown in Figure 6.  $\square$

Given  $2^n$  unitary maps  $\{U_a\}_{a=0}^{2^n-1}$ , it is immediate how to construct the direct sum  $\bigoplus_{a=0}^{2^n-1} U_a$  using controls on an ancilla of  $n$  qubits. This is illustrated in Figure 7. for the direct sum of  $2^3$  unitaries, and hopefully the ‘binary counting’ pattern on the control qubits is immediate.

Fig. 7. A circuit for the direct sum  $\bigoplus_{a=0}^7 U_a$



## 9 Quantum circuits for $U_M$

We now have sufficient background to give a solution to the problem posed in Section 3. Let us assume that  $\mathcal{M} = (Y, \mathcal{P}, S)$  is a frAM satisfying the conventions of Section 2.1, so

- (1)  $Y = \{0, \dots, 2^n - 1\}$ ,
- (2)  $S = \{0, \dots, 2^{n-1} - 1\}$ ,
- (3) The computation  $y \mapsto \{\mathcal{M}\}(y)$  terminates in under  $T = 2^t$  steps.

**Definition 17** *Throughout this section, the number of steps taken before termination on a given input will be important. Given  $y \in \{0, \dots, 2^{n-1} - 1\}$  as input to the classical machine  $\mathcal{M} = (Y, \mathcal{P}, S)$ , we define  $\tau(y)$ , the **number of steps to termination**, to be the minimal non-zero integer satisfying  $\mathcal{P}^{\tau(y)}(y) \in S$ . By (3) above,  $0 < \tau(y) < T$ , for all  $y \in \{0, \dots, 2^{n-1} - 1\}$ .*

Let us write the primitive evolution in matrix form, so

$$\mathcal{P} = \begin{pmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{pmatrix} : S \uplus W \rightarrow S \uplus W$$

We further assume that we have a quantum oracle  $U_{\mathcal{P}} = l_2(\mathcal{P})$  for the bijection  $\mathcal{P}$ . Using the description of the functor  $l_2$  provided in Section 7, we may therefore give  $U_{\mathcal{P}}$  as a block matrix, as

$$U_{\mathcal{P}} = \begin{pmatrix} l_2(P_{00}) & l_2(P_{01}) \\ l_2(P_{10}) & l_2(P_{11}) \end{pmatrix}$$

Note that each of the entries of this block matrix is a partial isometry. This is not true for matrix decompositions of *arbitrary* unitary maps (see Section 10 for more on this).

We may then give an explicit description of the oracle for  $\{\mathcal{M}\}$ , as follows:

**Proposition 18** *The unitary oracle  $U_{\mathcal{M}}$  for the bijection  $\{\mathcal{M}\}$  computed by*

$\mathcal{M}$  is given by

$$U_{\mathcal{M}} = l_2(\text{Res}_S(\mathcal{P})) = l_2(P_{00}) + \sum_{a=0}^{\infty} l_2(P_{01})l_2(P_{11})^a l_2(P_{10})$$

**PROOF.** This follows from the explicit algebraic description of the bijection  $\{\mathcal{M}\}$  given in Proposition 9, together with the properties of the  $l_2 : \mathbf{fpBij} \rightarrow \mathbf{Hilb}_{\mathbf{FD}}$  functor given in Theorem 13.  $\square$

We now give a circuit that implements this  $n - 1$  qubit unitary oracle, using the standard quantum circuit toolbox, a  $t + 1$  qubit ancilla, and calls to the oracles<sup>3</sup>  $U_{\mathcal{P}}$  and  $U_{\mathcal{P}}^{-1}$ .

### 9.1 An overview of the complete circuit

The circuit presented is naturally divided into three blocks, together with some elementary operations, as shown in Figure 9. For reasons of space, the details of blocks  $A, B$  and  $C$  are given in Appendix I, and explicit matrices for the unitaries implemented by each of these blocks are given in Appendix II.

In Figure 9., the  $X$  gate is simply the unconditional not-gate ( $X|0\rangle = |1\rangle$ ,  $X|1\rangle = |0\rangle$ ) and the “**Shift – R**” operation is the usual qubit-wise cyclic right-shift on a  $t + 1$  qubit register, defined on the computational basis by

$$|b_0 b_1 \dots b_t b_{t+1}\rangle \xrightarrow{\mathbf{Shift-R}} |b_{t+1} b_0 \dots b_{t-1} b_t\rangle$$

This has a simple, although not optimal, realisation via swap gates, as shown in Figure 8.

**Theorem 19** *Given an arbitrary superposition of computational basis states*

$$|\Psi\rangle = \sum_{a=0}^{2^{n-1}-1} \alpha_a |a\rangle$$

---

<sup>3</sup> At this point, we are following the convention that, when we have access to an oracle, we also have access to its inverse. For an oracle given as a quantum circuit, this is straightforward – in the general setting, this is very implementation-dependent.

Fig. 8. The qubit-wise cyclic right shift

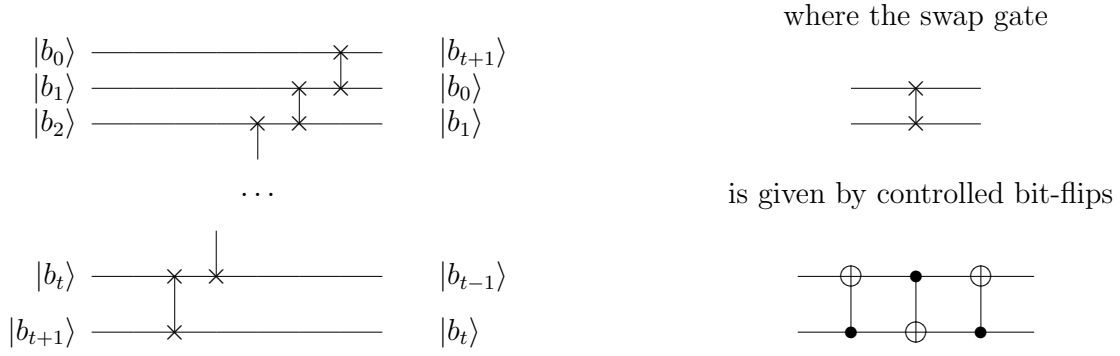
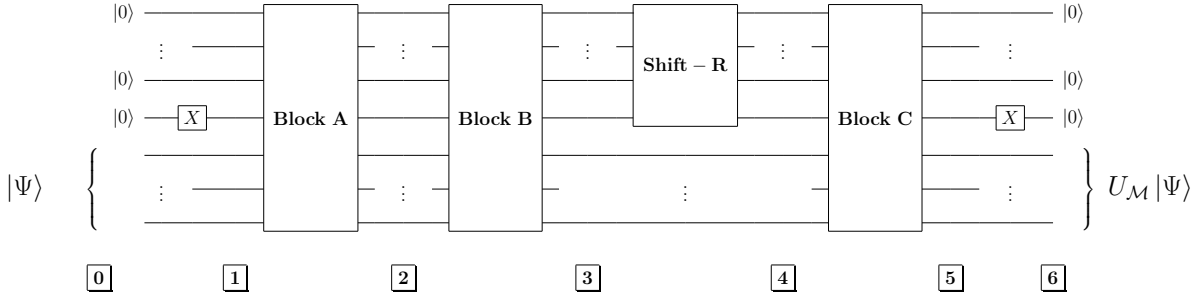


Fig. 9. The complete circuit for an oracle for  $\{\mathcal{M}\}$



then the circuit of Figure 9. acts as

$$|0\rangle |\Psi\rangle \mapsto |0\rangle \left( \sum_{a=0}^{2^{n-1}-1} \alpha_a |\{\mathcal{M}\}(a)\rangle \right)$$

i.e. this circuit provides an oracle for the bijection computed by  $\mathcal{M}$ .

The complete circuit requires  $3T$  calls to the oracle  $U_{\mathcal{P}}$  (or  $U_{\mathcal{P}}^{-1}$ ) for the primitive evolution of  $\mathcal{M}$ , where  $T$  is the upper bound to the number of steps before termination. This may readily be seen from the circuits given in Appendix I, where each of the three circuits presented requires exactly  $T$  calls to  $U_{\mathcal{P}}$ .

## PROOF.

We analyse the circuit of Figure 9. at the points 0/ to 6/ shown in this diagram. We consider two cases: where the input  $|\Psi\rangle$  is a computational basis vector  $|j\rangle$ , and where the input is an arbitrary superposition  $\sum_{a=0}^{2^{n-1}-1} \alpha_a |a\rangle$ .

Our claim is then that the action of the circuit in these two cases is as shown in Figure 10. In this Figure, recall that  $\tau(x)$  is the number of primitive steps

Fig. 10. The action of the Circuit of Figure 9.

	Computational Basis	Arbitrary Superposition
Point 0	$ \vec{0}\rangle  j\rangle$	$ \vec{0}\rangle \left( \sum_{a=0}^{2^{n-1}-1} \alpha_a  a\rangle \right)$
Point 1	$ \vec{1}\rangle  j\rangle$	$ \vec{1}\rangle \left( \sum_{a=0}^{2^{n-1}-1} \alpha_a  a\rangle \right)$
Point 2	$ \tau(j) - 1\rangle  \{\mathcal{M}\}(j)\rangle$	$\sum_{a=0}^{2^{n-1}-1} \alpha_a  \tau(a) - 1\rangle  \{\mathcal{M}\}(a)\rangle$
Point 3	$ 2\tau(j) - 2\rangle  j\rangle$	$\sum_{a=0}^{2^{n-1}-1} \alpha_a  2\tau(a) - 2\rangle  a\rangle$
Point 4	$ \tau(t) - 1\rangle  j\rangle$	$\sum_{a=0}^{2^{n-1}-1} \alpha_a  \tau(a) - 1\rangle  a\rangle$
Point 5	$ \vec{1}\rangle  \{\mathcal{M}\}(j)\rangle$	$\sum_{a=0}^{2^{n-1}-1} \alpha_a  \vec{1}\rangle  \{\mathcal{M}\}(a)\rangle$
Point 6	$ \vec{0}\rangle  \{\mathcal{M}\}(j)\rangle$	$ \vec{0}\rangle \left( \sum_{a=0}^{2^{n-1}-1} \alpha_a  \{\mathcal{M}\}(a)\rangle \right)$

from the start state  $x$  to the halt state  $\mathcal{M}(x)$ , for all  $x = 0, \dots, 2^{n-1} - 1$  (See Definition 9). We also use vector notation to distinguish the multi-qubit states  $|\vec{0}\rangle = |00 \dots 00\rangle$  and  $|\vec{1}\rangle = |00 \dots 01\rangle$  from the single-qubit states  $|0\rangle, |1\rangle$ .

To verify this, we now consider each step separately. Point 0 is a triviality - it is simply the definition of the input. The remaining steps are as follows:

- (1) On both the computational basis and the superposition inputs, this is straightforward. The  $X$ -gate flips the least significant bit of the ancilla, transforming the  $t + 1$  qubit state  $|0\rangle$  into the  $t + 1$  qubit state  $|1\rangle$ .
- (2) We consider the effect of block B on both computational basis elements, and on arbitrary superpositions separately, as follows:
  - **Computational Basis:** From Corollary 21 of Appendix II, block A implements the unitary map

$$\mathcal{A} = \begin{pmatrix} l_2(P_{01}) & l_2(P_{00}) & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ l_2(P_{01}P_{11}) & l_2(P_{01}P_{10}) & l_2(P_{00}) & 0 & 0 & \dots & 0 & 0 & 0 \\ l_2(P_{01}P_{11}^2) & l_2(P_{01}P_{11}P_{10}) & l_2(P_{01}P_{10}) & l_2(P_{00}) & 0 & \dots & 0 & 0 & 0 \\ l_2(P_{01}P_{11}^3) & l_2(P_{01}P_{11}^2P_{10}) & l_2(P_{01}P_{11}P_{10}) & l_2(P_{01}P_{10}) & l_2(P_{00}) & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ l_2(P_{01}P_{11}^{T-1}) & l_2(P_{01}P_{11}^{T-2}P_{10}) & l_2(P_{01}P_{11}^{T-3}P_{10}) & l_2(P_{01}P_{11}^{T-4}P_{10}) & l_2(P_{01}P_{11}^{T-5}P_{10}) & \dots & l_2(P_{01}P_{10}) & l_2(P_{00}) & 0 \\ l_2(P_{11}^T) & l_2(P_{11}^{T-1}P_{10}) & l_2(P_{11}^{T-2}P_{10}) & l_2(P_{11}^{T-3}P_{10}) & l_2(P_{11}^{T-4}P_{10}) & \dots & l_2(P_{11}P_{10}) & l_2(P_{10}) & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & I_{2^t} \end{pmatrix}$$

Note that each entry of this matrix is a  $(2^{n-1} \times 2^{n-1})$  block. As the ancilla of the input state is  $|\vec{1}\rangle = |00 \dots 01\rangle$ , we may simply read off the output state from column 1. of this matrix. As the input is in the computational basis, and the number of steps to termination is bounded we know that exactly one of the partial isometries from this column will give a non-zero answer when applied to  $|j\rangle$ . Thus, we are left with the computational basis state  $|\tau(j) - 1\rangle |\{\mathcal{M}(j)\}\rangle$ , by the explicit formula

for  $\{M\}$  given in Proposition 18.

- **Arbitrary Superposition:** This follows by the unitarity of block A, and the above result for a computational basis state. Given the product state  $|1\rangle (\sum_a \alpha_a |a\rangle) = \sum_a \alpha_a |1\rangle |a\rangle$ , block A will produce the highly entangled state  $\sum_a |\tau(a) - 1\rangle |\{M\}(a)\rangle$ .

At this point, we *almost* have the result we require (i.e. the output of an oracle for  $\{\mathcal{M}\}$ ). However, the result is entangled with the ancillary register — this entanglement is based on the number of steps before termination for the classical machine  $\mathcal{M} = (Y, \mathcal{P}, S)$  (this situation is as described in [34]).

Should we simply wish for the result of the computation, without this entanglement, we would be forced to trace out the ancillary register (using the standard linear algebra trace on the tensor product), causing decoherence. The remainder of the circuit is therefore devoted to a coherent way of disentangling the ancillary register with the result of the computation<sup>4</sup>.

- (3) Informally, the action of block B is simply to ‘uncompute’ the computation just performed — however, although the computation is undone, the ancilla that counts the number of computational steps continues to increment.

More formally (using analogous reasoning to (2) above), by Corollary 23, block B implements the following unitary map:

$$\mathcal{B} = \begin{pmatrix} l_2(P_{01})^\dagger & l_2(P_{00})^\dagger & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ l_2(P_{01}P_{11})^\dagger & l_2(P_{01}P_{10})^\dagger & l_2(P_{00})^\dagger & 0 & 0 & \dots & 0 & 0 & 0 \\ l_2(P_{01}P_{11}^2)^\dagger & l_2(P_{01}P_{11}P_{10})^\dagger & l_2(P_{01}P_{10})^\dagger & l_2(P_{00})^\dagger & 0 & \dots & 0 & 0 & 0 \\ l_2(P_{01}P_{11}^3)^\dagger & l_2(P_{01}P_{11}^2P_{10})^\dagger & l_2(P_{01}P_{11}P_{10})^\dagger & l_2(P_{01}P_{10})^\dagger & l_2(P_{00})^\dagger & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ l_2(P_{01}P_{11}^{T-1})^\dagger & l_2(P_{01}P_{11}^{T-2}P_{10})^\dagger & l_2(P_{01}P_{11}^{T-3}P_{10})^\dagger & l_2(P_{01}P_{11}^{T-4}P_{10})^\dagger & l_2(P_{01}P_{11}^{T-5}P_{10})^\dagger & \dots & l_2(P_{01}P_{10})^\dagger & l_2(P_{00})^\dagger & 0 \\ l_2(P_{11}^T)^\dagger & l_2(P_{11}^{T-1}P_{10})^\dagger & l_2(P_{11}^{T-2}P_{10})^\dagger & l_2(P_{11}^{T-3}P_{10})^\dagger & l_2(P_{11}^{T-4}P_{10})^\dagger & \dots & l_2(P_{11}P_{10})^\dagger & l_2(P_{10})^\dagger & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & I_{2^t} \end{pmatrix}$$

When applied to a computational basis vector of the form  $|p\rangle |j\rangle$ , this matrix gives the computational basis vector  $|p + \tau(j) - 1\rangle |\{M\}^{-1}(j)\rangle$  (provided  $p < T$ , and  $0 \leq j \leq 2^{n-1} - 1$ ). Thus, when applied to the computational basis vector  $|\tau(j) - 1\rangle |\{M\}(j)\rangle$ , matrix  $\mathcal{B}$  of Corollary 23 will produce

$$|2\tau(j) - 2\rangle |\{M\}^{-1}\{M\}(j)\rangle = |2\tau(j) - 2\rangle |j\rangle$$

<sup>4</sup> Readers familiar with reversible/quantum computation may wonder why we do not simply use a variation of Bennett’s ‘fan-out and uncompute’ technique to do this. This is addressed in Section 10.

Finally, when matrix  $\mathcal{B}$  of Corollary 23 is applied to a superposition of the form  $\sum_a \alpha_a |\tau(a) - 1\rangle |\{M\}(a)\rangle$ , linearity together with the above result implies that the result is  $\sum_a \alpha_a |2\tau(a) - 2\rangle |a\rangle$ .

- (4) The action of the **Shift – R** operation on the ancilla may be summarised, for *even* computational basis states, as  $|2k\rangle \mapsto |k\rangle$  (i.e. division by 2. Due to the form of the input at point **4**, the action on *odd* computational basis states is irrelevant). Thus a state of the form  $|2(k - 1)\rangle |\phi\rangle$  at point **4** is mapped to  $|k - 1\rangle |\phi\rangle$  at point **5**.

The required result is then immediate for computational basis states, and follows by linearity for superpositions.

- (5) Informally, block **C** re-performs the computation of Block **A**. However, in this case, the ancilla is *decremented* instead of incremented at each step. From Corollary 25 of Appendix II, block *C* implements the unitary map

$$\mathcal{C} = \begin{pmatrix} l_2(P_{10}) & l_2(P_{11}P_{10}) & \dots & l_2(P_{11}^{T-4}P_{10}) & l_2(P_{11}^{T-3}P_{10}) & l_2(P_{11}^{T-2}P_{10}) & l_2(P_{11}^{T-1}P_{10}) & l_2(P_{11}^T) & 0 \\ l_2(P_{00}) & l_2(P_{01}P_{10}) & \dots & l_2(P_{01}P_{11}^{T-5}P_{10}) & l_2(P_{01}P_{11}^{T-4}P_{10}) & l_2(P_{01}P_{11}^{T-3}P_{10}) & l_2(P_{01}P_{11}^{T-2}P_{10}) & l_2(P_{01}P_{11}^{T-1}) & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & l_2(P_{00}) & l_2(P_{01}P_{10}) & l_2(P_{01}P_{11}P_{10}) & l_2(P_{01}P_{11}^2P_{10}) & l_2(P_{01}P_{11}^3) & 0 \\ 0 & 0 & \dots & 0 & l_2(P_{00}) & l_2(P_{01}P_{10}) & l_2(P_{01}P_{11}P_{10}) & l_2(P_{01}P_{11}^2) & 0 \\ 0 & 0 & \dots & 0 & 0 & l_2(P_{00}) & l_2(P_{01}P_{10}) & l_2(P_{01}P_{11}) & 0 \\ 0 & 0 & \dots & 0 & 0 & 0 & l_2(P_{00}) & l_2(P_{01}) & 0 \\ 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & I_{2^t} \end{pmatrix}$$

Consider the action of this on some computational basis state  $|p\rangle |q\rangle$ , where  $T - 1 \leq p \leq 2T - 2$ , and  $0 \leq q \leq 2^{n-1} - 1$ . As the ancilla  $|p\rangle$  is a computational basis state, we may simply read off the result from the appropriate column. As  $|q\rangle$  is also a computational basis state, we know that exactly one of the partial isometries

$$l_2(P_{00}), l_2(P_{01}P_{10}), l_2(P_{01}P_{11}P_{10}), l_2(P_{01}P_{11}^2P_{10}), \dots, l_2(P_{01}P_{11}^{T-2}P_{10})$$

has a non-zero result when applied to  $|q\rangle$ , and so

$$\mathcal{C} |p\rangle |q\rangle = |p - (\tau(q) - 1)\rangle |\{M\}(q)\rangle$$

Thus, taking  $p = \tau(q) - 1$ , we deduce that  $\mathcal{C} |\tau(q) - 1\rangle |q\rangle = |1\rangle |\{M\}(q)\rangle$ . The result when applied to a superposition state follows by linearity, hence

$$\mathcal{C} \left( \sum_a \alpha_a |\tau(a) - 1\rangle |a\rangle \right) = \sum_a \alpha_a |1\rangle |\{M\}(a)\rangle = |1\rangle \left( \sum_a \alpha_a |\{M\}(a)\rangle \right)$$

as required. Note that this is a product state.

- (6) Here, the *X*-gate simply flips the least significant bit of the ancilla. Thus, the  $t + 1$  qubit state  $|1\rangle$  becomes the  $t + 1$  qubit state  $|0\rangle$ .

Putting these steps together, the circuit of Figure 9. acts as

$$|0\rangle |\Psi\rangle \mapsto |0\rangle \left( \sum_{a=0}^{2^n-1} \alpha_a |\{\mathcal{M}\}(a)\rangle \right)$$

as required.  $\square$

## 10 Using the scheme of Section 9 using arbitrary unitaries

Readers familiar with reversible and quantum computation will no doubt wonder why we used the scheme that we did in order to disentangle the ancilla from the result of the computation — a suitable scheme may be based on Bennett’s ‘fan-out and uncompute’ procedure [7,8]. Although our scheme may offer a slight reduction in the size on ancilla required, the overall time complexity is similar in both cases.

The justification for presenting the scheme in this way is that the circuit of Figure 9. is also applicable in certain cases where the oracle  $U_{\mathcal{P}}$  is replaced by a more general  $n$ -qubit unitary map  $U = \begin{pmatrix} U_{00} & U_{01} \\ U_{10} & U_{11} \end{pmatrix}$ . In this case, a computational basis input at point **0** of Figure 9. does *not* always give a computational basis state at point **3** — thus we cannot use a disentangling scheme based on fan-out.

We do not claim that this scheme always produces a product state with ancilla  $|0\rangle$ . However, when the resolution formula

$$Res_S(U) = U_{00} + \sum_{j=0}^{\infty} U_{01} U_{11}^j U_{10}$$

gives a unitary map, this scheme may be seen to produce a product state output.

However, the conditions for  $Res_S(U)$  to define a unitary map, when  $U$  is not simply a permutation of the computational basis, are less straightforward. Consider the following two very similar quantum logic gates:

- **Hadamard gate**

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

- **Square root of NOT**

$$HX = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}$$

Using the standard formula for the sum of a geometric series,

$$\text{Res}_S(H) = \frac{1 + \sqrt{2}}{2\sqrt{2}} \quad \text{and} \quad \text{Res}_S(HX) = \frac{1 - \sqrt{2}}{\sqrt{2} - 1} = -1$$

Considering these as maps on the complex plane

$$z \mapsto \left( \frac{1 + \sqrt{2}}{2\sqrt{2}} \right) z \quad \text{and} \quad z \mapsto -z$$

it is immediate that  $\text{Res}_S(HX)$  is unitary, whereas  $\text{Res}_S(H)$  is not.

A general characterisation of when the above formula defines a unitary map is complicated by the fact that for a general unitary, the matrix components  $U_{00}, U_{01}, U_{10}, U_{11}$  are *not* in general partial isometries [24]. We refer interested readers to [25] for the general case, where the ‘primitive evolution’ may be an arbitrary unitary (rather than simply an oracle for a classical permutation), and the number of time-steps allowed may be unbounded.

## 11 Acknowledgements

The author wishes to emphasise that many of the constructions of this paper (especially the general form of the matrices presented in Appendix II and their connection with the Resolution or categorical trace, and algebraic program semantics generally) are special cases of a more general theory [25] being developed in collaboration with Phil Scott (Ottawa), who has given permission for their use in this specific application.

The author is also very grateful to Sam Braunstein (York) for assistance in coming to terms with the quantum circuit paradigm, and many useful discussions on both quantum computation generally and the problems raised by conditional iteration in quantum computing in particular, and Manas Patra (York) for the connection between period-finding, and upper bounds for termination described in Appendix III.

The author also wishes to thank Samson Abramsky and Bob Coecke (Oxford) for many discussions on categorical traces, category-theoretic approaches

to quantum mechanics, and game semantics. Thanks are due to Keye Martin (U.S. Naval Research Laboratories) for promoting a domain-theoretic approach to Abstract Machines and their computations – this approach made the classical results of Sections 1.-6. possible.

Finally, many thanks to the anonymous referees of Theoretical Computer Science, for very helpful suggestions on improving readability and presentation of this paper.

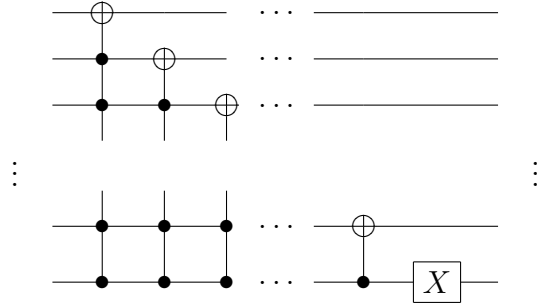
## References

- [1] S. Abramsky and R. Jagadeesan: Games and full completeness for multiplicative linear logic *Journal of Symbolic Logic* 59 543-574 (1994)
- [2] S. Abramsky: Retracing some paths in Process algebra, In *CONCUR 96, Lecture Notes in Computer Science*, pp.1-17, Springer-Verlag (1996)
- [3] S. Abramsky and B. Coecke: A categorical semantics of quantum protocols. In *Proc. 19th Annual IEEE Symp. on Logic in Computer Science (LICS 2004)*, IEEE Computer Soc. Press, 415-425 (2005)
- [4] S. Abramsky: Abstract Scalars, Loops, and Free Traced and Strongly Compact Closed Categories, in *Proceedings of CALCO 2005, Springer LNCS 3629*, 1–31 (2005)
- [5] S. Abramsky, E. Haghverdi and P. Scott: Geometry of Interaction and Linear Combinatory Algebras, *Mathematical Structures in Computer Science* 12(5) (2002)
- [6] M. Barr: Algebraically Compact Functors *Journal of Pure and Applied Algebra* 82 Elsevier 211-231 (1992)
- [7] C. Bennett: Logical Reversibility of Computation *IBM Journal of Research and Development* (17) (1979)
- [8] C. Bennett: Time-space trade-offs for reversible computation *SIAM J. Comput.* 18 766 (1989)
- [9] E. Bernstein and U. Vazirani: Quantum Complexity Theory *SIAM Journal on Computing* 26(5) 1411-1473 (1997)
- [10] A. Blass: A game semantics for linear logic *Annals of Pure and Applied Logic* 56 151-166 (1992)
- [11] V. Danos and L. Regnier: Local and asynchronous  $\beta$ -reduction, *Proceedings of the Eighth Annual IEEE Symp. on Logic in Computer Science* (1993)
- [12] D. Deutsch: Quantum theory, the Church-Turing principle and the universal quantum computer *Proc. Royal Society of London, A* 400 (1985) 97-117

- [13] T. Draper: Addition on a Quantum Computer, *arXiv:quant-ph 0008033v1* (2000)
- [14] J.-Y. Girard: Geometry of Interaction I, Proceedings Logic Colloquium '88, North-Holland 221-260 (1988)
- [15] E. Haghverdi: *A categorical approach to linear logic, geometry of proofs and full completeness*, PhD thesis, U. of Ottawa, (2000)
- [16] E. Haghverdi and P. Scott: A categorical model for the Geometry of Interaction, *Automata, Languages and Programming: Proc. 31st International Colloquium, Springer LNCS 2004*, (2004) Journal version : [17]
- [17] E. Haghverdi and P. Scott: A categorical model for the Geometry of Interaction, *Theoretical Computer Science 350(2)* 252-274 (2006)
- [18] E. Haghverdi and P. Scott: Geometry of Interaction and the Dynamics of Proof Reduction: a tutorial, in *New Structures for Physics, Springer Lecture Notes in Physics series* (to appear)
- [19] M. Hasegawa: On traced monoidal closed categories *Math. Struct. Comp. Sci.* (to appear)
- [20] P. Hines: *The Algebra of Self-Similarity and its Applications*, PhD Thesis, U. of Wales, Bangor (1998)
- [21] P. Hines: Physical systems as constructive logics, in *Calude, C., Dinneen, M., Paun, G., Rozenberg, G., Stepney, S. (editors). Unconventional Computation, UC 2006: York, UK.* Springer. LNCS (2006)
- [22] P. Hines: Machine Semantics - from causality to computational models, *International Journal of Unconventional Computation 4(3)* (2008) 249-272
- [23] P. Hines: Machine Semantics, *Theoretical Computer Science 409* (2008), 1-23.
- [24] P. Hines and S. Braunstein: The structure of partial isometries, in *Semantic Techniques in Quantum Computation, Cambridge University Press* (to appear)
- [25] P. Hines and P. Scott: Categorical traces from single-photon linear optics (*Submitted*) available as <http://www.peterhines.net/downloads/papers/optics.pdf>
- [26] P. Hines: From termination to cyclic behaviour: machine semantics and period-finding, (*in preparation*) Drafts available from: <http://www.peterhines.net/downloads/drafts/cyclic-termination/>
- [27] A. Joyal, R. Street and D. Verity: Traced Monoidal categories, *Math. Proc. Camb. Phil. Soc.* 425-446 (1996)
- [28] G. Kelly and M. Laplaza: Coherence for compact closed categories, *Journal of Pure and Applied Algebra 19* 193-213 (1980)
- [29] T. Kieu and M. Danos: A No-Go Theorem for Halting a Universal Quantum Computer *Acta Physica Hungarica A 14(1)* 217-225 (2001)

- [30] M.V. Lawson: *Inverse Semigroups* World Scientific, Singapore (1998)
- [31] Y. Lecerf: Machine de Turing réversible. Insolubilité récursive en  $n \in \mathbb{N}$  de l'équation  $u=\theta^n$  où  $\theta$  est un "isomorphism de codes", *Comptes Rendus* 257 2579-2600 (1963)
- [32] R. Levine and A. Sherman: A note on Bennett's time-space trade-off for reversible computation, *SIAM Jour. Comp.* 19(4) 673-677 (1990)
- [33] M. Li and P. Vitanyi: Reversible simulation of irreversible computation, *Proc. 11th IEEE Conference on Computational Complexity* 301-306 (1996)
- [34] N. Linden and S. Popsecue: The halting problem for quantum computers, *arXiv:quant-ph:/9806054 v2* (1998)
- [35] C. Lutz: JANUS – a time-reversible language. *Letter to Landauer, available as: <http://www.cise.ufl.edu/~mpf/rc/janus.html>* (1986)
- [36] E. Manes and M. Arbib: *Algebraic Approaches to Program Semantics* Springer-Verlag (1986)
- [37] R. van Meter and K. Itoh: Fast quantum modular exponentiation *Phys. Rev. A* (71) (2005)
- [38] T. Miyadera and M. Ohya: Designing Quantum Turing Machine is Uncomputable, *Proc. EQIS02* (2002)
- [39] T. Miyadera and M. Ohya: On Halting Process of Quantum Turing Machine, *Open Systems and Information Dynamics* 12(3) 261-265 (2005)
- [40] J. Myers: Can a Universal Computer be Fully Quantum? *Physical Review Letters* 78 (9) (1997) 1823-1824
- [41] M. Nielsen and I. Chuang: *Quantum Computation and Quantum Information*, Cambridge University Press (2000)
- [42] P. Selinger: Towards a quantum programming language *Mathematical Structures in Computer Science* 14(4) (2004) 527-586
- [43] P. Shor: Polynomial-time algorithms for prime factorisation and discrete logarithms on a quantum computer *SIAM review* 41 303-332 (1999)
- [44] V. Vedral and M. Plenio: Basics of Quantum Computation *arXiv:quant-ph/9802065v1* (1998)

Fig. 11. The modular successor function

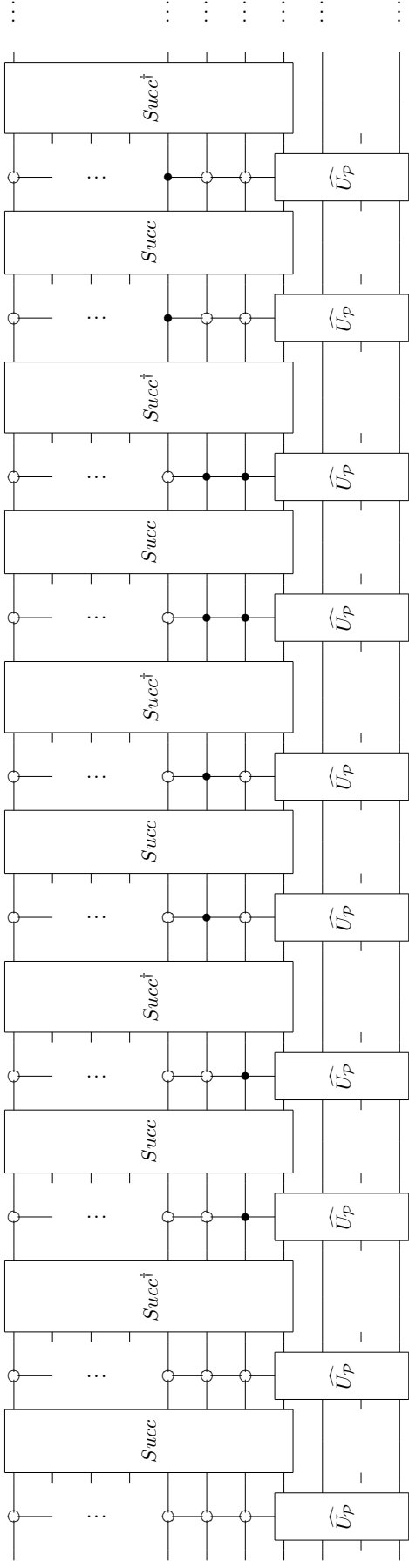


## Appendix I: circuits for blocks A, B, and C

We now present circuits for blocks A,B and C of Figure 9. These circuits require the modular successor function on  $t + 1$  qubits, defined on the computational basis by  $Succ |k\rangle = |k + 1 \pmod{2^{t+1}}\rangle$ . We give a naive circuit for this in Figure 11. For a more sophisticated approach, we refer to either [44] for elementary arithmetic operations, or [13] for a neat use of the quantum Fourier transform to implement modular addition and other arithmetic operations.

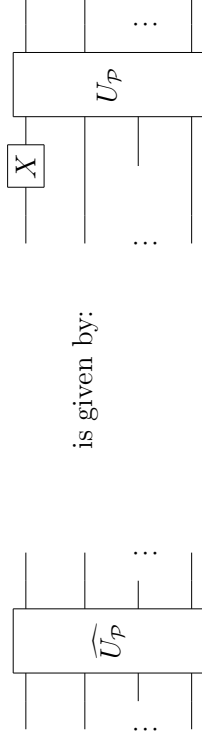
Note also that in the following circuit blocks, the quantum oracle  $U_{\mathcal{P}}$  (or indeed its inverse  $U_{\mathcal{P}}^{-1}$ ) is always used in conjunction with a quantum not-gate applied to the most significant bit (i.e. the start-halt qubit). This has the effect of interchanging the start-halt subspace and the working subspace. Although the justification for the following circuits is simply that the corresponding matrix manipulations give us the right answers, readers seeking a deeper motivation are referred to the field of *game semantics for linear logic* [10,1], where repeated interchange of ‘player’ and ‘opponent’ (or ‘prover’ and ‘disprover’) forms a crucial element of the system (see [14] for similar ideas, motivating this field).

**Block A** The circuit for Block A in Figure 9. is given by:



Where:

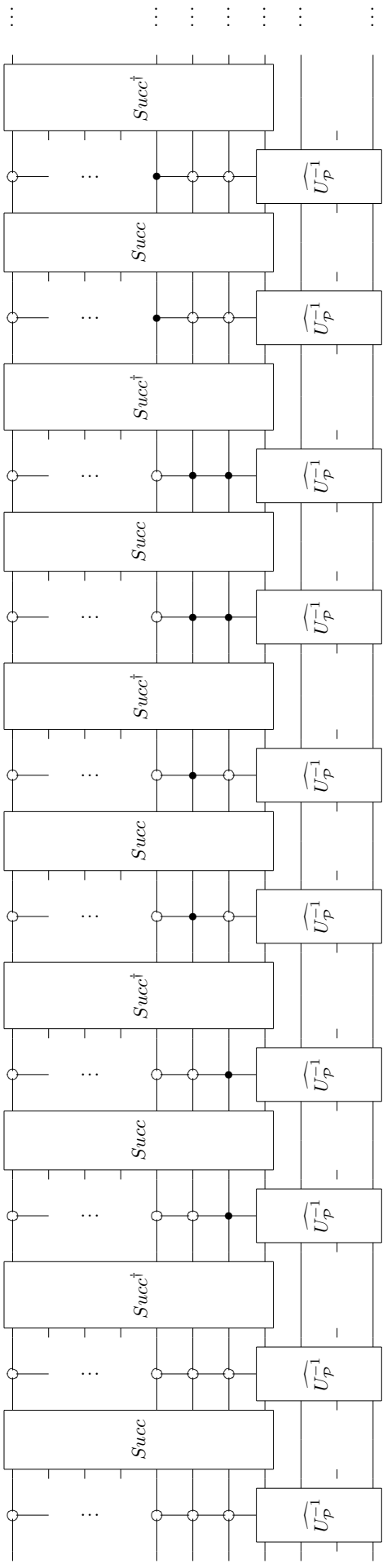
- The operator  $\widehat{U}_{\mathcal{P}}$  is derived from the oracle  $U_{\mathcal{P}}$  as follows:



- $Succ$  denotes the  $t + 2$  qubit successor operation, and  $Succ^\dagger$  is its inverse.
- A total of  $T$  controlled operations are applied.

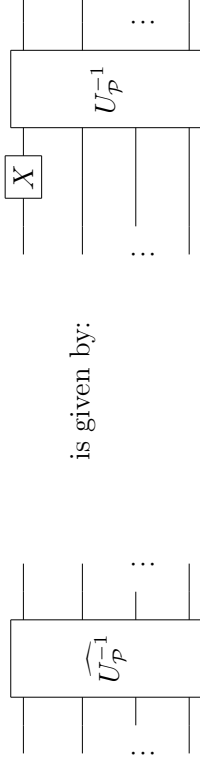
**Parsing the above circuit:** Note that in the above circuit, every second controlled operation is conjugated by the modular successor gate, as shown. Thus, the final operation in this circuit is the  $Succ$  operation, and *not* a controlled  $U_{\widehat{\mathcal{P}}}$ .

**Block B** The circuit for block B in Figure 9. is very similar to that of block A:



Where:

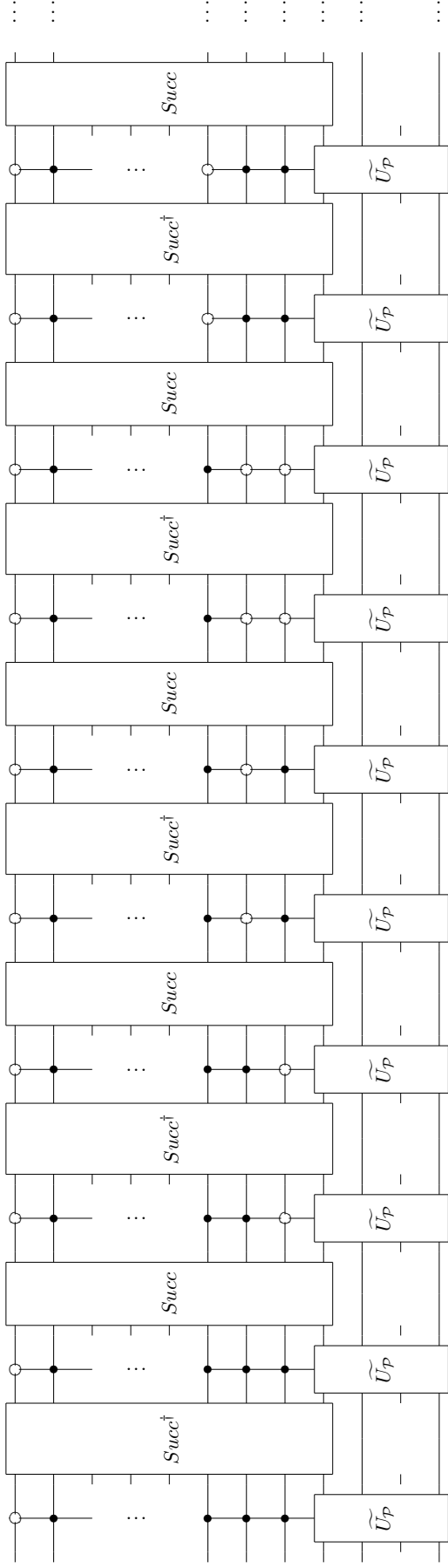
- The operator  $\widehat{U_P^{-1}}$  is derived from the inverse of the oracle  $U_P$  as follows:



- $Succ$  denotes the  $t + 2$  qubit successor operation, and  $Succ^\dagger$  is its inverse.
- A total of  $2T$  controlled operations are applied.

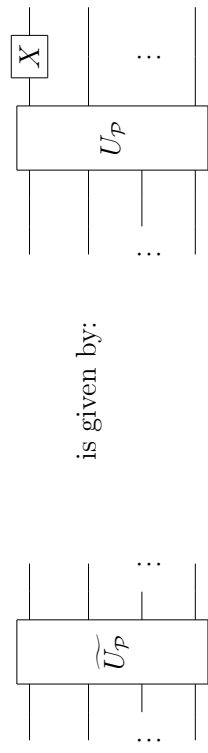
**Parsing the above circuit:** Note that in the above circuit, every second controlled operation is again conjugated by the modular successor operation. The final operation in this circuit is again the  $Succ$  operation.

**Block C** The circuit for Block C of Figure 9. is similar to that of block A, with the order of the control on the ancilla reversed:



Where:

- The operator  $\widetilde{U}_P$  is derived from the oracle  $U_P$  as follows:



- $Succ$  denotes the  $t + 2$  qubit successor operation, and  $Succ^\dagger$  is its inverse.
- A total of  $T$  controlled operations are applied.

**Parsing the above circuit:** In the above circuit, every second controlled operation is again (as in Blocks A and B) conjugated by the modular successor operation.

## Appendix II: Matrices for blocks A, B and C

### A matrix for block A

**Proposition 20** *Block A of Figure 9. produces the unitary map*

$$\mathcal{A} = G_{T-1}G_{T-2}\dots G_1G_0$$

where the matrix  $G_x$  is given in terms of

$$U_{\mathcal{P}} = \begin{pmatrix} l_2(P_{00}) & l_2(P_{01}) \\ l_2(P_{10}) & l_2(P_{11}) \end{pmatrix}$$

as follows:

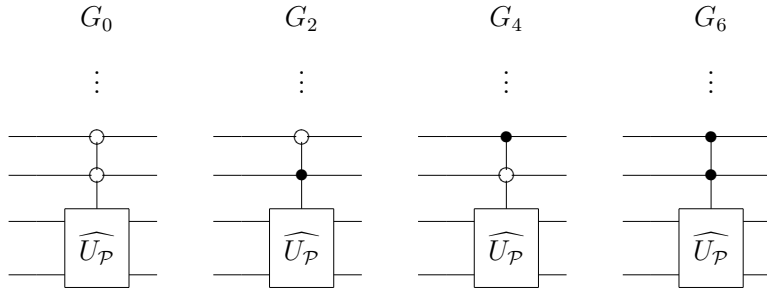
$$G_x = \begin{pmatrix} I_{2^{n-1-x}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & l_2(P_{01}) & l_2(P_{00}) & \mathbf{0} \\ & l_2(P_{11}) & l_2(P_{10}) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & I_{T2^n - x2^{n-1}} \end{pmatrix} \quad \text{where } x = 0, \dots, T-1$$

**PROOF.** First note that the circuit for  $\widehat{U}_{\mathcal{P}}$  modifies the oracle for primitive evolution  $U_{\mathcal{P}}$  by flipping the most significant qubit (ie. the start-halt qubit) of the output of  $U_{\mathcal{P}}$ . Therefore,

$$\widehat{U}_{\mathcal{P}} = \begin{pmatrix} l_2(P_{01}) & l_2(P_{00}) \\ l_2(P_{11}) & l_2(P_{10}) \end{pmatrix}$$

This then allows us to give  $G_x$  in terms of controlled operations on  $\widehat{U}_{\mathcal{P}}$ .

When  $x$  is even,  $G_x$  is given by the connection between direct sum representations and controlled gates given in Section 8. We may build up  $G_0, G_2, G_4, \dots$ , as shown below:



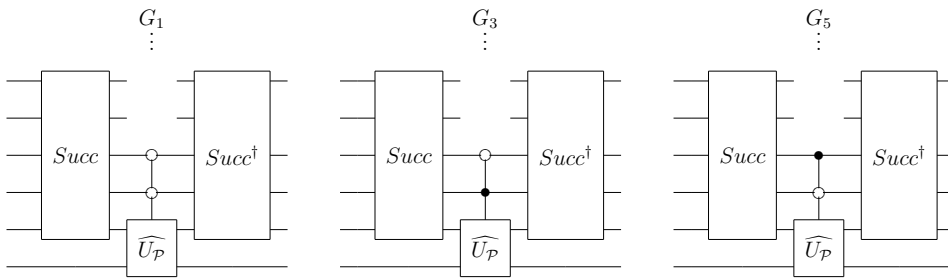
Let us now conjugate one of the above controlled operations by the  $t+1$  qubit (modular) successor function,  $Succ$ . This is applied to the ancillary register, together with the most significant qubit of  $\widehat{U}_{\mathcal{P}}$ . When a matrix of the form

$$G_x = \begin{pmatrix} I_{2^{n-1}x} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \widehat{U}_{\mathcal{P}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & I_{T_{2^n-x}2^{n-1}} \end{pmatrix}$$

is conjugated by this operation, the result is

$$\begin{pmatrix} I_{2^{n-1}(x+1)} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \widehat{U}_{\mathcal{P}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & I_{T_{2^n-(x+1)}2^{n-1}} \end{pmatrix}$$

However, this is exactly  $G_{x+1}$ , as required. Thus we may build up  $G_1, G_3, G_5, \dots$  from controlled operations and the successor function, as shown:



□

**Corollary 21** *An explicit matrix for the unitary map  $\mathcal{A}$  implemented by block  $A$  is*

$$A = \begin{pmatrix} l_2(P_{01}) & l_2(P_{00}) & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ l_2(P_{01}P_{11}) & l_2(P_{01}P_{10}) & l_2(P_{00}) & 0 & 0 & \dots & 0 & 0 & 0 \\ l_2(P_{01}P_{11}^2) & l_2(P_{01}P_{11}P_{10}) & l_2(P_{01}P_{10}) & l_2(P_{00}) & 0 & \dots & 0 & 0 & 0 \\ l_2(P_{01}P_{11}^3) & l_2(P_{01}P_{11}^2P_{10}) & l_2(P_{01}P_{11}P_{10}) & l_2(P_{01}P_{10}) & l_2(P_{00}) & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ l_2(P_{01}P_{11}^{T-1}) & l_2(P_{01}P_{11}^{T-2}P_{10}) & l_2(P_{01}P_{11}^{T-3}P_{10}) & l_2(P_{01}P_{11}^{T-4}P_{10}) & l_2(P_{01}P_{11}^{T-5}P_{10}) & \dots & l_2(P_{01}P_{10}) & l_2(P_{00}) & 0 \\ l_2(P_{11}^T) & l_2(P_{11}^{T-1}P_{10}) & l_2(P_{11}^{T-2}P_{10}) & l_2(P_{11}^{T-3}P_{10}) & l_2(P_{11}^{T-4}P_{10}) & \dots & l_2(P_{11}P_{10}) & l_2(P_{10}) & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & I_{2^t} \end{pmatrix}$$

**PROOF.** We prove this by induction:

First note that, by direct calculation, and the functoriality of  $l_2$ ,

$$G_1G_0 = \begin{pmatrix} l_2(P_{01}) & l_2(P_{00}) & \mathbf{0} & \mathbf{0} \\ l_2(P_{01}P_{11}) & l_2(P_{01}P_{10}) & l_2(P_{00}) & \mathbf{0} \\ l_2(P_{11}^2) & l_2(P_{11}P_{10}) & l_2(P_{10}) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & I \end{pmatrix}$$

Now assume that for some  $k < T - 1$ ,

$$G_kG_{k-1}\dots G_0 = \begin{pmatrix} l_2(P_{01}) & l_2(P_{00}) & 0 & 0 & \dots & 0 & 0 \\ l_2(P_{01}P_{11}) & l_2(P_{01}P_{10}) & l_2(P_{00}) & 0 & \dots & 0 & 0 \\ l_2(P_{01}P_{11}^2) & l_2(P_{01}P_{11}P_{10}) & l_2(P_{01}P_{10}) & l_2(P_{00}) & \dots & 0 & 0 \\ l_2(P_{01}P_{11}^3) & l_2(P_{01}P_{11}^2P_{10}) & l_2(P_{01}P_{11}P_{10}) & l_2(P_{01}P_{10}) & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ l_2(P_{01}P_{11}^{k-1}) & l_2(P_{01}P_{11}^{k-2}P_{10}) & l_2(P_{01}P_{11}^{k-3}P_{10}) & l_2(P_{01}P_{11}^{k-4}P_{10}) & \dots & l_2(P_{00}) & 0 \\ l_2(P_{11}^k) & l_2(P_{11}^{k-1}P_{10}) & l_2(P_{11}^{k-2}P_{10}) & l_2(P_{11}^{k-3}P_{10}) & \dots & l_2(P_{10}) & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & I \end{pmatrix}$$

Direct calculation, and the functoriality of  $l_2$  gives that

$$G_{k+1}G_kG_{k-1}\dots G_0 = \begin{pmatrix} l_2(P_{01}) & l_2(P_{00}) & 0 & 0 & 0 & \dots & 0 & 0 \\ l_2(P_{01}P_{11}) & l_2(P_{01}P_{10}) & l_2(P_{00}) & 0 & 0 & \dots & 0 & 0 \\ l_2(P_{01}P_{11}^2) & l_2(P_{01}P_{11}P_{10}) & l_2(P_{01}P_{10}) & l_2(P_{00}) & 0 & \dots & 0 & 0 \\ l_2(P_{01}P_{11}^3) & l_2(P_{01}P_{11}^2P_{10}) & l_2(P_{01}P_{11}P_{10}) & l_2(P_{01}P_{10}) & l_2(P_{00}) & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ l_2(P_{01}P_{11}^k) & l_2(P_{01}P_{11}^{k-1}P_{10}) & l_2(P_{01}P_{11}^{k-2}P_{10}) & l_2(P_{01}P_{11}^{k-3}P_{10}) & l_2(P_{01}P_{11}^{k-4}P_{10}) & \dots & l_2(P_{00}) & 0 \\ l_2(P_{11}^{k+1}) & l_2(P_{11}^kP_{10}) & l_2(P_{11}^{k-1}P_{10}) & l_2(P_{11}^{k-2}P_{10}) & l_2(P_{11}^{k-3}P_{10}) & \dots & l_2(P_{10}) & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & I \end{pmatrix}$$

Our result thus follows by induction.  $\square$

Now note the similarity between the columns of this matrix (excluding the

first column<sup>5</sup>), and the summands of the resolution of  $\mathcal{P} : X \rightarrow X$ ,

$$\text{Res}_S(\mathcal{P}) = P_{00} + P_{01}P_{10} + P_{01}P_{11}P_{10} + P_{01}P_{11}^2P_{10} + \dots$$

(or rather, their image under the functor  $l_2$ ). This is the key to its action in the proof of Theorem 19.

## A matrix for block B

**Proposition 22** *Block B of Figure 9. produces the unitary map*

$$\mathcal{B} = H_{2T-2}H_{2T-3} \dots H_1H_0$$

where the matrix  $H_x$  is given in terms of  $U_{\mathcal{P}}^{-1} = \begin{pmatrix} l_2(P_{00})^\dagger & l_2(P_{10})^\dagger \\ l_2(P_{01})^\dagger & l_2(P_{11})^\dagger \end{pmatrix}$  by:

$$H_x = \begin{pmatrix} I_{2^{n-1}x} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & l_2(P_{10})^\dagger l_2(P_{00})^\dagger & \mathbf{0} & \mathbf{0} \\ & l_2(P_{11})^\dagger l_2(P_{01})^\dagger & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & I_{T2^n - x2^{n-1}} \end{pmatrix} \quad \text{where } x = 0, \dots, 2(T-1)$$

**PROOF.** Note that the circuit for  $\widehat{U_{\mathcal{P}}^{-1}}$  modifies the oracle for the inverse of the primitive evolution  $U_{\mathcal{P}}$  by flipping the most significant qubit (ie. the start-halt qubit) of the output of  $U_{\mathcal{P}}$ . Therefore,

$$\widehat{U_{\mathcal{P}}} = \begin{pmatrix} l_2(P_{10}^{-1}) & l_2(P_{00}^{-1}) \\ l_2(P_{11}^{-1}) & l_2(P_{01}^{-1}) \end{pmatrix}$$

The proof that  $H_x$  may be built up as shown in terms of controlled operations on  $\widehat{U_{\mathcal{P}}^{-1}}$  is then almost identical to the proof of Proposition 20.  $\square$

**Corollary 23** *An explicit matrix for the unitary map  $\mathcal{B}$  implemented by block B is*

<sup>5</sup> Readers familiar with the field of Algebraic Program Semantics will recognise column 0. as providing the summands of a construction known as the Elgot Dagger [36] — the usual program semantics representation of a “While” loop. An analysis of this is beyond the scope of this paper.

$$\mathcal{B} = \begin{pmatrix} l_2(P_{01})^\dagger & l_2(P_{00})^\dagger & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ l_2(P_{01}P_{11})^\dagger & l_2(P_{01}P_{10})^\dagger & l_2(P_{00})^\dagger & 0 & 0 & \dots & 0 & 0 & 0 \\ l_2(P_{01}P_{11}^2)^\dagger & l_2(P_{01}P_{11}P_{10})^\dagger & l_2(P_{01}P_{10})^\dagger & l_2(P_{00})^\dagger & 0 & \dots & 0 & 0 & 0 \\ l_2(P_{01}P_{11}^3)^\dagger & l_2(P_{01}P_{11}^2P_{10})^\dagger & l_2(P_{01}P_{11}P_{10})^\dagger & l_2(P_{01}P_{10})^\dagger & l_2(P_{00})^\dagger & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ l_2(P_{01}P_{11}^{T-1})^\dagger & l_2(P_{01}P_{11}^{T-2}P_{10})^\dagger & l_2(P_{01}P_{11}^{T-3}P_{10})^\dagger & l_2(P_{01}P_{11}^{T-4}P_{10})^\dagger & l_2(P_{01}P_{11}^{T-5}P_{10})^\dagger & \dots & l_2(P_{01}P_{10})^\dagger & l_2(P_{00})^\dagger & 0 \\ l_2(P_{11}^T)^\dagger & l_2(P_{11}^{T-1}P_{10})^\dagger & l_2(P_{11}^{T-2}P_{10})^\dagger & l_2(P_{11}^{T-3}P_{10})^\dagger & l_2(P_{11}^{T-4}P_{10})^\dagger & \dots & l_2(P_{11}P_{10})^\dagger & l_2(P_{10})^\dagger & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & I_{2^t} \end{pmatrix}$$

**PROOF.** The proof of this is sufficiently close to that of Corollary 21 not to need reproducing separately. Simply replace the matrix

$$\widehat{U}_{\mathcal{P}} = \begin{pmatrix} l_2(P_{01}) & l_2(P_{00}) \\ l_2(P_{11}) & l_2(P_{10}) \end{pmatrix}$$

in the proof of Corollary 21 by the matrix

$$\widehat{U}_{\mathcal{P}}^{-1} = \begin{pmatrix} l_2(P_{10}^{-1}) & l_2(P_{00}^{-1}) \\ l_2(P_{11}^{-1}) & l_2(P_{01}^{-1}) \end{pmatrix}$$

The inductive step (and hence the full result) follows immediately.  $\square$

### A matrix for block C

**Proposition 24** *Block C of Figure 9. produces the unitary map*

$$\mathcal{C} = K_0 K_1 \dots K_{T-2} K_{T-1}$$

where the matrix  $K_x$  is given in terms of

$$U_{\mathcal{P}}^{-1} = \begin{pmatrix} l_2(P_{00})^\dagger & l_2(P_{10})^\dagger \\ l_2(P_{01})^\dagger & l_2(P_{11})^\dagger \end{pmatrix}$$

as follows:

$$K_x = \begin{pmatrix} I_{2^{n-1}x} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & l_2(P_{10})^\dagger & l_2(P_{11})^\dagger & \mathbf{0} \\ & l_2(P_{00})^\dagger & l_2(P_{01})^\dagger & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & I_{T2^{n-x}2^{n-1}} \end{pmatrix} \quad \text{where } x = T-1, \dots, 0$$

**PROOF.** The circuit for  $\widetilde{U}_{\mathcal{P}}$  modifies the oracle for the inverse of the primitive evolution  $U_{\mathcal{P}}$  by flipping the most significant qubit (ie. the start-halt qubit) of the output of  $U_{\mathcal{P}}$ . However, by contrast to blocks A and B, this happens *after*  $U_{\mathcal{P}}$  is applied, rather than before. Direct calculation gives

$$\widetilde{U}_{\mathcal{P}} = \begin{pmatrix} l_2(P_{10}) & l_2(P_{11}) \\ l_2(P_{00}) & l_2(P_{01}) \end{pmatrix}$$

The proof that  $K_x$  may be built up as shown in terms of controlled operations on  $\widetilde{U}_{\mathcal{P}}$  is again almost identical to the proof of Proposition 20.  $\square$

**Corollary 25** *An explicit matrix for the unitary map  $\mathcal{C}$  implemented by block  $\mathcal{C}$  is*

$$\mathcal{C} = \begin{pmatrix} l_2(P_{10}) & l_2(P_{11}P_{10}) & \dots & l_2(P_{11})^{T-4}P_{10} & l_2(P_{11}^{T-3}P_{10}) & l_2(P_{11}^{T-2}P_{10}) & l_2(P_{11}^{T-1}P_{10}) & l_2(P_{11}^T) & 0 \\ l_2(P_{00}) & l_2(P_{01}P_{10}) & \dots & l_2(P_{01}P_{11}^{T-5}P_{10}) & l_2(P_{01}P_{11}^{T-4}P_{10}) & l_2(P_{01}P_{11}^{T-3}P_{10}) & l_2(P_{01}P_{11}^{T-2}P_{10}) & l_2(P_{01}P_{11}^{T-1}) & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & l_2(P_{00}) & l_2(P_{01}P_{10}) & l_2(P_{01}P_{11}P_{10}) & l_2(P_{01}P_{11}^2P_{10}) & l_2(P_{01}P_{11}^3) & 0 \\ 0 & 0 & \dots & 0 & l_2(P_{00}) & l_2(P_{01}P_{10}) & l_2(P_{01}P_{11}P_{10}) & l_2(P_{01}P_{11}^2) & 0 \\ 0 & 0 & \dots & 0 & 0 & l_2(P_{00}) & l_2(P_{01}P_{10}) & l_2(P_{01}P_{11}) & 0 \\ 0 & 0 & \dots & 0 & 0 & 0 & l_2(P_{00}) & l_2(P_{01}) & 0 \\ 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & I_{2^s} \end{pmatrix}$$

**PROOF.** We again prove this by induction.

Direct calculation and functoriality gives that

$$K_{T-2}K_{T-1} = \begin{pmatrix} I & 0 & 0 & 0 & 0 \\ 0 & l_2(P_{10}) & l_2(P_{11})l_2(P_{10}) & l_2(P_{11})^2 & 0 \\ 0 & l_2(P_{00}) & l_2(P_{01})l_2(P_{10}) & l_2(P_{01})l_2(P_{11}) & 0 \\ 0 & 0 & l_2(P_{00}) & l_2(P_{10}) & 0 \\ 0 & 0 & & 0 & I \end{pmatrix} = \begin{pmatrix} I & 0 & 0 & 0 & 0 \\ 0 & l_2(P_{10}) & l_2(P_{11}P_{10}) & l_2(P_{11}^2) & 0 \\ 0 & l_2(P_{00}) & l_2(P_{01}P_{10}) & l_2(P_{01}P_{11}) & 0 \\ 0 & 0 & l_2(P_{00}) & l_2(P_{10}) & 0 \\ 0 & 0 & & 0 & I \end{pmatrix}$$

(where the size of the identity blocks is implicit from the context).

Now assume that, for some  $T - 1 \geq s > 0$ ,

$$K_s \dots K_{T-2} K_{T-1} = \begin{pmatrix} I & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & I & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & l_2(P_{10}) & l_2(P_{11}P_{10}) & \dots & l_2(P_{11})^{K-4}P_{10} & l_2(P_{11}^{K-3}P_{10}) & l_2(P_{11}^{K-2}P_{10}) & l_2(P_{11}^{K-1}P_{10}) & l_2(P_{11}^K) & 0 \\ 0 & 0 & l_2(P_{00}) & l_2(P_{01}P_{10}) & \dots & l_2(P_{01}P_{11}^{K-5}P_{10}) & l_2(P_{01}P_{11}^{K-4}P_{10}) & l_2(P_{01}P_{11}^{K-3}P_{10}) & l_2(P_{01}P_{11}^{K-2}P_{10}) & l_2(P_{01}P_{11}^{K-1}) & 0 \\ 0 & 0 & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & l_2(P_{00}) & l_2(P_{01}P_{10}) & l_2(P_{01}P_{11}P_{10}) & l_2(P_{01}P_{11}^2P_{10}) & l_2(P_{01}P_{11}^3) & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & l_2(P_{00}) & l_2(P_{01}P_{10}) & l_2(P_{01}P_{11}P_{10}) & l_2(P_{01}P_{11}^2) & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & l_2(P_{00}) & l_2(P_{01}P_{10}) & l_2(P_{01}P_{11}) & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & l_2(P_{00}) & l_2(P_{01}) & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & I_{2^t} \end{pmatrix}$$

Direct calculation gives that  $K_{s+1}K_s \dots K_{T-2}K_{T-1} =$

$$\begin{pmatrix} I & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & l_2(P_{10}) & l_2(P_{11}P_{10}) & l_2(P_{11}^2) & \dots & l_2(P_{11})^{K-3}P_{10} & l_2(P_{11}^{K-2}P_{10}) & l_2(P_{11}^{K-1}P_{10}) & l_2(P_{11}^K) & l_2(P_{11}^{K+1}) & 0 \\ 0 & l_2(P_{00}) & l_2(P_{01}P_{10}) & l_2(P_{01}P_{11}P_{10}) & \dots & l_2(P_{01}P_{11}^{K-4}P_{10}) & l_2(P_{01}P_{11}^{K-3}P_{10}) & l_2(P_{01}P_{11}^{K-2}P_{10}) & l_2(P_{01}P_{11}^{K-1}P_{10}) & l_2(P_{01}P_{11}^K) & 0 \\ 0 & 0 & l_2(P_{00}) & l_2(P_{01}P_{10}) & \dots & l_2(P_{01}P_{11}^{K-5}P_{10}) & l_2(P_{01}P_{11}^{K-4}P_{10}) & l_2(P_{01}P_{11}^{K-3}P_{10}) & l_2(P_{01}P_{11}^{K-2}P_{10}) & l_2(P_{01}P_{11}^K) & 0 \\ 0 & 0 & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & l_2(P_{00}) & l_2(P_{01}P_{10}) & l_2(P_{01}P_{11}P_{10}) & l_2(P_{01}P_{11}^2P_{10}) & l_2(P_{01}P_{11}^3) & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & l_2(P_{00}) & l_2(P_{01}P_{10}) & l_2(P_{01}P_{11}P_{10}) & l_2(P_{01}P_{11}^2) & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & l_2(P_{00}) & l_2(P_{01}P_{10}) & l_2(P_{01}P_{11}) & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & l_2(P_{00}) & l_2(P_{01}) & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & I_{2^t} \end{pmatrix}$$

Our result therefore follows by induction.  $\square$

### Appendix III: finding an upper bound for termination by quantum period-finding

A spectacularly successful technique in quantum computation is *period-finding*, with the most celebrated example being Shor's factorisation algorithm [43], based on period-finding for modular exponential functions. We now demonstrate how in certain cases, quantum period-finding may be useful for finding an upper bound for the number of steps before termination of a frAM.

**Definition 26** *Given a bijection on a finite set  $f : X \rightarrow X$ , the **period** of  $F_x(n) = f^n(x)$  is the smallest non-zero integer  $K$  such that  $f_x(K+r) = f_x(r)$ , for all  $r \in \mathbb{N}$ . Equivalently, the period is the smallest  $K$  such that  $f^K(x) = x$ . It is immediate that such an integer exists, by the finiteness of  $X$ .*

Efficient quantum algorithms for finding such periods of such bijections may be found in (for example) [41] p. 241, as an application of the quantum Fourier transform, and the abelian hidden subgroup problem, using exactly one call to an oracle for  $F_x$ .

**Lemma 27** *Let  $\mathcal{M} = (X, \mathcal{P}, S)$  be a frAM. Then the period  $K$  of  $p_s(n) =$*

$\mathcal{P}^n(s)$  is an upper bound  $T$  for the number of steps before termination in the computation of  $\{\mathcal{M}\}(s)$ .

**PROOF.** Consider a start-halt configuration  $s \in S \subseteq S$ . Then by definition of the period  $K$ ,  $\mathcal{P}^K(s) = s$ , and so  $\mathcal{P}(s) \in S$ . Therefore,  $K \geq T$ .  $\square$

Thus, given an arbitrary superposition of starting states  $\sum_{j=0}^r \alpha_j |s_j\rangle$  we may calculate an appropriate upper bound for use in our algorithm using exactly  $r$  calls to a suitable oracle. Note that there are numerous subtleties associated with this – for example, although the period provides an upper bound for the number of steps before termination, it does not (except for certain carefully-crafted cases, e.g. abstract machines for modular exponentiation) provide the least upper bound. Similarly, the naive scheme presented above appears to require classical knowledge of the distinct branches in a superposition. The author conjectures that this is not, in fact, an essential requirement: a detailed study is work in progress [26].