Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science

Electronic Notes in Theoretical Computer Science 270 (1) (2011) 191–210

www.elsevier.com/locate/entcs

Quantum Circuits: From a Network to a One-Way Model

Larisse Voufo[1]

Institute for Scientific Interchange Foundation, 10133 Torino, Italy

Department of Computer Science, Indiana University, Bloomington, IN 47405

Abstract

We present a translation from the standard network model of quantum computation to the one-way model of quantum computation. The translation is compositional, i.e., it preserves the structure of computations, which allows us to abstract the concrete realizations in a monadic abstraction layer.
We briefly review the By-Product and Measurement Calculus approaches to combining circuits in the one-way model, and show how the procedures can be expressed in the exact same notation, bearing a direct relationship with the representation of circuits in the standard network model, and using monads. Discussions on improving the abstraction yield us to introducing an alternative approach to combining circuits in the one-way model: a graphical one.

Keywords: Computation, Monads, Measurement Calculus, By-Product, Graphical

1 Introduction

There are several models and implementations of quantum computation, most notably: quantum Turing machines and automata, quantum circuit (or network) models, adiabatic quantum computation, measurement-based (one-way) quantum computation, and topological quantum computation. Although all these models have been shown to be equivalent, their presentation typically "looks different." In particular, a given computation in the circuit model is likely to look rather different — structurally — when expressed in the one-way model.

In this paper, we study two of these models of computation in detail: the standard network (SN) model and the one-way (OW) model [11,3]. We study how computations are expressed in each of these two models and how to translate from one model to the other in a compositional manner, i.e., while preserving the structure of the computation.

[1] Email: lvoufo@cs.indiana.edu

doi:10.1016/j.entcs.2011.01.018

Once defined, the compositional translation exposes common structures that are abstracted using the mathematical construction of a monad. In more detail, we provide an abstract model of quantum computation which can, with a choice of two parameters, be instantiated to either the SN model or the OW model. The construction gives an elegant way to translate from one model to the other, and potentially enables optimizations expressed in one model to be easily transferred to the other model. Throughout the translation, we pay close attention to its efficiency and ability to be systematic.

In addition, the construction allows us to introduce an alternative approach to reasoning about circuits in the OW model that is both graphical and allows to easily approximate circuits.

In the following, we assume a basic familiarity with notions of quantum mechanics and computation, fundamental differences between the circuit and the OW models, as well as with the relation of OW realizations to graphs. [2] In addition, for simplicity, we will refer to the Pauli matrices σ_x, σ_y and σ_z as X, Y and Z, respectively. We will also precede any controlled operation with the letter C.

2 Composition in the One-Way Model

We begin with a review of the OW model of computation with a focus on *composition*. More precisely, we are interested in understanding the various ways in which two OW computations can be combined to produce a larger OW computation.

2.1 Elementary One-Way Computations

The premise of the OW model is to drive the computation by performing several one-qubit measurements (most of them in parallel), on a highly entangled state (ideally given by nature). Often, due to the non-deterministic nature of measurements, the process results in some temporary state, encoding the desired result in a Pauli basis. To uncover that result, one thus needs to perform what is known as classical post-processing, which consists of applying the appropriate Pauli operators.

For example, take the realization of a Hadamard – $H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$, and given an input $|\psi\rangle = \begin{pmatrix} a \\ b \end{pmatrix}$ (with $a^2 + b^2 = 1$), compute $H\,|\psi\rangle$. Given an initial state preparation that entangles [3] $|\psi\rangle$ (qubit 1) with some ancilla qubit $|+\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ (qubit 2), i.e., resulting in,

$$\begin{aligned} CZ\ (|\psi\rangle \otimes |+\rangle) &= CZ\ \left((a\,|0\rangle + b\,|1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right) \\ &= \frac{(a+b)\,|+\rangle + (a-b)\,|-\rangle}{\sqrt{2}}\,|0\rangle + \frac{(a-b)\,|+\rangle + (a+b)\,|-\rangle}{\sqrt{2}}\,|1\rangle \qquad (1) \end{aligned}$$

one simply needs to measure qubit 1 in the X-basis — signaling an outcome s_1 of 0 (or 1) if the measurement had collapsed the state into the positive (or negative)

[2] For more information, please see references [7], [10,9] and [1].

[3] Through some Ising-type interaction simulated by CZ operations

eigenstate $|+\rangle$ (or $|-\rangle$) of X.
This propagates the information from the logical input – qubit 1, to the logical output – qubit 2, so that, if $s_1 = 1$, then one must apply the Pauli X onto the result in qubit 2. To put it another way, one must apply X^{s_1} on qubit 2 after the measurement of qubit 1.

So, from Eq.(1), we get: $$\begin{cases} |+\rangle \otimes \left(\frac{a+b}{\sqrt{2}}|0\rangle + \frac{a-b}{\sqrt{2}}|1\rangle\right) & \text{with probability } \frac{1}{2} \quad \text{and } s_1 = 0 \\ |-\rangle \otimes \left(\frac{a-b}{\sqrt{2}}|0\rangle + \frac{a+b}{\sqrt{2}}|1\rangle\right) & \text{with probability } \frac{1}{2} \quad \text{and } s_1 = 1 \end{cases}$$
after the measurement. Then, the correction brings us to $\frac{a+b}{\sqrt{2}}|0\rangle + \frac{a-b}{\sqrt{2}}|1\rangle = H|\psi\rangle$.

Typically, to indicate the qubit acted upon by a given operation, one subscripts the operation with the index of the qubit. And to distinguish a reference to a logical qubit, from that to a general one (including ancillae), one simply surrounds the index of the logical qubit with braces. For instance, we say that the H-procedure described above realizes the unitary $H_{[1]}$ to indicate that the logical input value is initially stored in qubit 1. In addition, while $X_2^{s_1}$ indicates the correction on the actual qubit 2, $X_{[2]}^{s_1}$ indicates a correction on the logical output that qubit 2 corresponds to.

When it comes to composing circuits, there are several approaches that vary based on the understanding and expression of the OW procedure we just described above. Nevertheless, they all share a common understanding: Circuits compositions must be standardized before being applied to the input.

2.2 Composition

Take the composition of two H's into an Identity $I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, that is, attempting to realize $I_{[1]} = H_{[2]} \cdot H_{[1]}$.

Taken straightforwardly, this applies one H-procedure after another, making the output from the first application become the input into the second. Notably, it interleaves sequences of entanglement, measurement, and correction operations. Even though the process returns a valid result, it is not the behavior we desire.

Instead, by the definition of the OW procedure, we want the overall procedure for I to be in some standard form that has all entanglements coming first, followed by all measurements, and then followed by all corrections.

This standardization process differs from one approach to the other. From current literature, we find two main such approaches: The original – Raussendorf *et al.*'s *By-Product* (BP) [11], and its automated generalization – Danos *et al.*'s *Measurement Calculus* (MC) [3].

2.2.1 The By-Product Approach

This approach seems mostly mathematically grounded. It tends to reason about OW circuits in terms of the unitaries they realize before the corrections have been applied, and in association with the necessary entangled state preparation followed by the sequence of measurements. It then calls the conjugate transpose of the corrections that have yet to be performed "*by-product*" operators.

For example, for the H-procedure above, the realized unitary of concern is

$$\overset{ow}{H}_{[1]} = X_{[2]}^{s_1} \cdot H_{[1]} \ ;$$

where $X_{[2]}^{s_1}$ is the BP operator. Then, the associated preparation is constituted of the initial entanglement between qubits 1 and 2 (after their tensor product); while the measurement pattern is constituted of the single measurement M_1^x. (Note that a measurement is typically super-scripted with an indication of the basis it is measuring in. In this case, we are measuring in the X-basis.)

To construct the realization for I, one needs to standardize

$$\overset{ow}{I}_{[1]} = X_{[3]}^{s_2} \cdot H_{[2]} \cdot X_{[2]}^{s_1} \cdot H_{[1]} \qquad \text{into a form} \qquad \overset{ow}{I}_{[1]} = \tilde{I} \cdot I_{[1]} \ ;$$

where $\tilde{I}$ is the BP operator.

Concurrently, one must also reorganize the sequence of preparations and measurements, from both sub-circuits, consistently.

By some principles of graph state properties and algebraic manipulations (e.g. the stabilizer formalism), we are able to propagate the previous BP operator $X_{[2]}^{s_1}$ over the subsequent unitary $H_{[2]}$, and find $\overset{ow}{I}_{[1]} = X_{[3]}^{s_2} \cdot Z_{[2]}^{s_1} \cdot H_{[2]} \cdot H_{[1]} \ ;$ yielding the BP operator $\tilde{I} = X_3^{s_2} \cdot Z_3^{s_1}$.

Concurrently, we find we can simply combine the preparations and measurements from the sub-circuits directly. Thus, for I, the preparation consists of the entanglement of qubits 1 and 2, then of that of qubits 2 and 3. Meanwhile, the measurements consist of those of qubits 1 and then 2, in the X-basis.

As a general observation, since the overall composition and standardization (C&S) process here manipulates unitaries, it is not simple to systematically implement it in a computer program.

To make matters worse, non-Clifford operators [4] , i.e. those operators that do not map (sequences of) Pauli operators to one another, add complexity, as they cause the resulting sequences of measurements to have to be revisited. Essentially, their presence in a composition causes the final realized unitary to be different from that desired, by some set of defining parameters. And as a result, one must substitute those defining parameters in accordingly in order to recover the desired unitary.

The overall procedure has time complexity quadratic in the number of subcircuits and linear in the input size.

Things get simpler than this under the next approach.

2.2.2 The Measurement Calculus Approach

This approach defines a number of commands to express each entanglement, measurement, and correction in a sequence. We will be referring to such sequence as

[4] For instance, take a general unitary rotation $U_{rot}(\alpha,\beta,\zeta) = U_x(\zeta) \cdot U_z(\beta) \cdot U_x(\alpha)$, with $U_x(\alpha) = e^{-i\frac{\alpha}{2}X}$ and $U_z(\alpha) = e^{-i\frac{\alpha}{2}Z}$.
When mapped to Pauli operators, it undergoes some modifications according to the following:

$$\begin{cases} U_{rot}(\alpha,\beta,\zeta) \cdot X = & X \cdot U_{rot}(\alpha,-\beta,\zeta) \\ U_{rot}(\alpha,\beta,\zeta) \cdot Z = & Z \cdot U_{rot}(-\alpha,\beta,-\zeta) \end{cases} .$$

Clearly, for some parameter values – such as $\alpha = \zeta = 0$ and $\beta = \pm\frac{\pi}{2}$ for a $\pm\frac{\pi}{2}Phase = \begin{bmatrix} 1 & 0 \\ 0 & \pm i \end{bmatrix}$, either $U_{rot}(\alpha,\beta,\zeta) \neq U_{rot}(\alpha,-\beta,\zeta)$ or $U_{rot}(\alpha,\beta,\zeta) \neq U_{rot}(-\alpha,\beta,-\zeta)$.

(i) $E_{ij}X_i^s \longrightarrow X_i^s Z_j^s E_{ij}$

(ii) $E_{ij}Z_i^s \longrightarrow Z_i^s E_{ij}$

(iii) $E_{ij}A_{\boldsymbol{k}} \longrightarrow A_{\boldsymbol{k}}E_{ij}$, with $A \neq E$

(iv) ${}^t[M_i^\alpha]^s X_i^r \longrightarrow {}^t[M_i^\alpha]^{s+r}$

(v) ${}^t[M_i^\alpha]^s Z_i^r \longrightarrow {}^{t+r}[M_i^\alpha]^s$

(vi) $A_{\boldsymbol{k}}X_i^s \longrightarrow X_i^s A_{\boldsymbol{k}}$,
with $A \neq X$ and $A \neq Z$

(vii) $A_{\boldsymbol{k}}Z_i^s \longrightarrow Z_i^s A_{\boldsymbol{k}}$, with $A \neq X$ and $A \neq Z$

(viii) ${}^t[M_i^\alpha]^s \longrightarrow S_i^t[M_i^\alpha]^s$

(ix) $X_j^s S_i^t \longrightarrow S_i^t X_j^{s[t+s_i/s_i]}$

(x) $Z_j^s S_i^t \longrightarrow S_i^t Z_j^{s[t+s_i/s_i]}$

(xi) ${}^t\left[M_j^\alpha\right]^s S_i^r \longrightarrow S_i^r{}^{t[r+s_i/s_i]}\left[M_j^\alpha\right]^{s[r+s_i/s_i]}$

(xii) $\perp S \longrightarrow \perp$

Figure 1. Standardization passes.

"*CmdSeq*". For now, let us describe its structure in more detail.

We have already seen the measurement command in the form of M_q^α (q represents a qubit). An extension of this representation accounts for cases where some angle $\alpha' = (-1)^s\alpha + \pi t$, with $s, t \in \mathbb{Z}$. Then, the measurement is represented as: $M_q^{\alpha'} = {}^t[M_q^\alpha]^s$.

Correction commands are of the form X_q^s we saw earlier, except that one can also use Z in place of X. For entanglement, a notation E_{pq} indicates the entanglement between two qubits p and q.

There is also a *shift* command S_q^s used to express and manipulate dependencies incurred by applying Z-corrections before measurements.

The command sequence is part of a larger representation for an entire procedure — a.k.a. *patterns*, that includes the set of qubits (V), and logical inputs (I) and outputs (O):

$$(V,\ I,\ O,\ CmdSeq) \ .$$

For example, the H-procedure above is now represented as $(\{1,2\},\ \{1\},\ \{2\},\ X_2^{s_1}M_1^x E_{12})$.

Here, the standardization into I is now a matter of applying a set of symmetric passes, such as the ones in Fig. 1, on the composition of the command sequences for the two H-operations involved. $X_3^{s_2}M_2^x E_{23}X_2^{s_1}M_1^x E_{12}$.

Concurrently, the remaining components of the pattern also needs to be updated accordingly.

After Passes i, iii, iv, and vii, we obtain the following as the pattern for I:

$$(\{1,2,3\},\ \{1\},\ \{3\},\ X_3^{s_2}Z_3^{s_1}M_2^x M_1^x E_{23}E_{12}) \ .$$

In a general perspective, the MC process simplifies the BP approach out of the effects from non-Clifford operators (cf. end of Section 2.2.1). A basic procedure is described in Alg. 1. Essentially, Steps 1 and 2 of the new procedure constitute its core part, while Step 3 is optional; since it only serves in simplifying patterns, by shifting dependencies induced by Z-corrections out of consideration when specifying measurements (cf. Passes v and x).

Not accounting for Step 3, the time complexity is linear in the number n of sub-circuits, and cubic in the input size s. The inclusion of Step 3 makes it quadratic in $(n * s)$.

So, to recap, no matter the choice of approach, during a composition step, we

Algorithm 1 A Basic C&S Procedure.

Standardizing $C_2M_2E_2C_1M_1E_1$ into some form $CME = \tilde{C}_2\tilde{\tilde{\tilde{C}}}_1\tilde{\tilde{M}}_2\tilde{M}_1E_2E_1$, with each $\sim$-decoration indicating a change in the pattern.

```
1. Propagate each element of $E_2$ backwards, across $C_1$      $\longrightarrow E = E_2 \cdot E_1 \ ; \ \tilde{C}_1$
2. Propagate each element of $M_2$ backwards, across $\tilde{C}_1$      $\longrightarrow M = \tilde{M}_2 \cdot M_1 \ ; \ \tilde{\tilde{C}}_1 \ ; \ C = C_2 \cdot \tilde{\tilde{C}}_1$
3. For each $M$:
   a. Introduce shift      $\longrightarrow CS\tilde{M}E$
   b. Propagate the shift forward, across the remaining part of $M$ and $C$,
      dropping it at the end of the command sequence.      $\longrightarrow \tilde{C}\tilde{M}E$
```

E_x, M_x, and C_x respectively represent the sequence of entanglement, measurement and Pauli correction operations of two standard sub-patterns 1 and 2, with x indicating the sub-pattern that is acted upon.

have a notion of two given sub-realizations R_i and R_j that need to be combined into a larger realization R. The process requires that the composition of the sub-realizations be standardized before we can move on. Particular to each approach are both the representations for the realizations and the way that the C&S procedure is performed. Overall, the MC approach addresses the efficiency and systematization needs of our translation better.

However, as the sizes of circuits increase, these patterns can quickly become too tedious to manipulate, or simply reason about, at a high level. For instance, simply consider generating the command sequence for a CCZ operation, when it is expressed in terms of controlled rotations. Fig. A.1 in the Appendix (A) gives us a pretty good idea.

Consequently, one has to wonder whether there would not be a better way to represent these patterns (and any model-specific quantum computation in general), in a way that hides some implementation details and mostly reason about the circuits in terms of input/output.

As it so happens, this is very possible if one defines an appropriate monad for computations in the OW model. In a broader spectrum, an abstraction using this concept of monads provides a unified framework for understanding and reasoning about quantum computation in general, across differing models.

We are now moving on to present our monadic abstraction.

3 A Unifying Framework for Quantum Computation

Again, take the composition of two H-operations into a I-operation.

3.1 The Representation

In quantum mechanics, under the vector-state approach,

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = H \cdot H \ .$$

That is, given a quantum state $|\psi\rangle$, one can obtain the same state after two consecutive applications of H: $I\,|\psi\rangle = H\,(H\,|\psi\rangle)$.

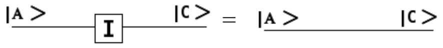
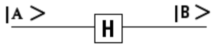
We express this as

$$\begin{aligned} I\ (A) \quad = \quad & H\ (A) \quad \gg= \quad \backslash(B) \quad \rightarrow \\ & H\ (B) \quad \gg= \quad \backslash(C) \quad \rightarrow \\ & return\ (C) \end{aligned} \tag{2}$$

If A represents the logical input, then B represents the logical output of the first application of H. Then, B becomes the logical input of the second application, and C represents the logical output of the entire operation. Naturally, both A and C refer to the same value. The sequencing is expressed with the arrows $\rightarrow$. We will define $\gg=$ and *return* shortly.

The application of an operation to a given state is a matrix-vector multiplication. And two consecutive applications correspond to a matrix multiplication.

The SN model provides a diagrammatic perspective for understanding and interpreting quantum mechanics. Here, matrices are now represented by some labeled squares, while the vectors acted upon are quantum registers represented by lines. For example, with time flowing from left to right, the operations we are working with are now represented as follows:

$I =$  and $H =$  .

It is common practice to label each line with a name for the register or an indication of the value it holds at some given time [5] .

To combine operations, or multiply their matrices, one simply connects output registers from one operation to input registers into another one, as in Fig. 2 for our ongoing example.

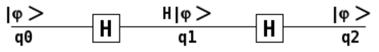


Figure 2. Identity as Two Consecutive Hadamards, in the SN model

Our abstract expression is certainly the same as in Eq. (2), even though our underlying representation has changed.

Under the OW model, our abstract expression is still the same as in Eq. (2). However, due to the need for the standardization of results, one needs to remember previous realizations at each sequential step; which makes things a little bit trickier. To better illustrate this, let us first generalize the abstraction, by presenting the concepts of using monads.

3.2 The Unified Framework: The Monads

In Eq. (2), $\gg=$(*bind*) and *return* are, in fact, monadic operations that abstract the representation of computations away from their implementation or meaning; encapsulating each possible implementation (or meaning) in a so-called "monad". While we maintain the exact same notation across different models, the model-specific implementation details are specified by simply changing the underlying monad.

[5] For multiple registers, it suffices to display multiple lines in parallel, and apply individual or collective operations on them. Varying other conventions are used to represent controlled operations, some special operations such as negations, and operations over multiple registers. Please see Chapter 3 of [10] for a general example.

In earlier work, A. Sabry [12], then Vizzotto *et al.* [14] used these operations (as well as *arrow*-constructors), to relate "unusual" features of quantum computing, i.e. quantum parallelism and measurement, to monads (and *arrows*). Herein, we exploit the idea to unify differing approaches to reasoning about quantum circuits, giving an approach that centers around considering logical inputs and outputs of operations solely.

Before we move further, it helps to briefly describe the concept of a monad. Since we are using its semantic constructions from the theory of Programming Languages, we will be defining it from that perspective.[6]

3.2.1 Brief Review

Basically, the idea is as follows:

- Take an element of type $\mathbf{A}$, and define some "context" $\mathbf{m}$ for it. Within the context, elements are now of type $\mathbf{m\,A}$.

  For example, in our case herein, we take a classical **bit**, and the context of the bit is its "quantumness", hence defining a **qubit**. Type $\mathbf{A}$ is $\mathbf{Bit}$, whereas $\mathbf{m\,A}$ is $\mathbf{Qubit}$.

- *return* lifts given elements into some monadic layer. In other words, it takes an element of type $\mathbf{A}$ and returns a corresponding element of type $\mathbf{m\,A}$. We say that *return* is of type $\mathbf{A} \to \mathbf{m\,A}$.

  In our case, the type is $\mathbf{Bit} \to \mathbf{Qubit}$, and this means producing the quantum equivalent of given classical bits. So,

  $$\mathit{return}\ 0 = |0\rangle,\ \mathit{return}\ 1 = |1\rangle, \text{ and more generally } \quad \mathit{return}\ A = |A\rangle.$$

- $\gg\!=$(*bind*), on some input of type $\mathbf{m\,A}$ and special function f of type $\mathbf{A} \to \mathbf{m\,B}$, binds f to the input elements in a way that simulates the application of some quantum function f^*, of type $\mathbf{m\,A} \to \mathbf{m\,B}$. We say that $\gg\!=$ is of type $\mathbf{m\,A} \to (\mathbf{A} \to \mathbf{m\,B}) \to \mathbf{m\,B}$, and $\exists f^* = (\gg\!= f), \forall f$.

  For example, to simulate the application of a H-matrix in the SN model – as taking in a vector and returning another vector, the H used in our abstraction (Eq. (2)) is defined as follows – taking in a bit and returning a vector:

  $$\begin{cases} H\,0 &= \quad |+\rangle \\ H\,1 &= \quad |-\rangle \end{cases} . \tag{3}$$

  Concurrently, $\gg\!=$ may be defined as follows[7] :

  $$\begin{pmatrix} a \\ b \end{pmatrix} \gg\!= f = (a \text{ `*' } (f\ 0)) \text{ `+' } (b \text{ `*' } (f\ 1)) . \tag{4}$$

  From Eqs. (3) and (4), applying $(\gg\!= H)$ on input $|\psi\rangle$ is equivalent to multiplying it with the corresponding matrix since

  $$(a\,|+\rangle \text{ `+' } b\,|-\rangle) = \frac{1}{\sqrt{2}} \begin{pmatrix} a+b \\ a-b \end{pmatrix} = H\,|\psi\rangle .$$

[6] A more mathematically-based definition of a monad is provided by Category theory.
[7] '$*$' denotes a scalar product, and '$+$' a vector addition.

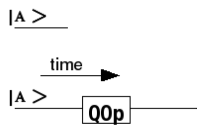
Q-mechanics	Diagrammatically
$return\ A \ = \ \lvert A\rangle$ $\lvert A\rangle \gg= Op \ = \ QOp \cdot \lvert A\rangle$	$return\ A \ =$ $\lvert A\rangle \gg= Op \ =$ 

Figure 3. Monadic Operations for the SN model.

- Importantly, there are three laws that must be satisfied in order to ensure the consistency of computations that are based on these monads:
  · Left-identity: $ma \gg= return \ = \ ma$
  · Right-identity: $return\ x \gg= f \ = \ f\ x$
  · Associativity: $(ma \gg= f) \gg= g \ = \ ma \gg= (\backslash a \to (f\ a) \gg= g)$

  Now, we can clearly describe our abstraction.

3.2.2 The Monadic Abstraction

For our abstraction herein, we basically have *return* lifting an element A of the computational basis (classical) to the quantum layer – i.e. to $|A\rangle$. However, whenever appropriate, it can lift the information further to a model-specific quantum layer, e.g. the OW model. (We will come back to the latter shortly.)

For now, to each expression $|\mathbf{A}\rangle \gg= \backslash\mathbf{A} \to \mathbf{Op\ A}$ — or equivalently $|\mathbf{A}\rangle \gg= \mathbf{Op}$, we associate an operation $\mathbf{QOp}$ that simulates the underlying matrix multiplication as in Eq. (4). Then, based on this association, we illustrate how one could define the monadic operations for the SN model in Fig. 3.

The implementations resulting from such abstraction constitute a level of implementation details hidden away by the abstraction. In reality, these details are often tedious to manipulate, as per our earlier consideration for expressing the CCZ operation under the MC approach to the OW model. But, given this abstraction, we can now express it simply as in Fig. 4.

$$\boxed{\begin{aligned}
CCZ\ (A0, B0, C0) \ &= \\
C\tfrac{\pi}{2}Phase\ (B0, C0) \ &\gg= \ \backslash(B1, C1) \to \\
CNot\ (A0, B1) \ &\gg= \ \backslash(A0, B2) \to \\
C(-\tfrac{\pi}{2})Phase\ (B2, C1) \ &\gg= \ \backslash(B3, C2) \to \\
CNot\ (A0, B3) \ &\gg= \ \backslash(A0, B4) \to \\
C\tfrac{\pi}{2}Phase\ (A0, C2) \ &\gg= \ \backslash(A1, C3) \to \\
return\ (A1, B4, C3) &
\end{aligned}}$$

Figure 4. CCZ – Monadic Abstraction.

On that note, let us now take a look at the OW model, and define its monadic layer as well.

3.2.3 The One-Way Model

We start with an expression

$$|A\rangle \gg= Op_i \ ; \tag{5}$$

where $(\gg= Op_i)$ corresponds to the application of some OW realization R_i — and some operation QOp_i. This is trivial, as one can just go ahead and apply the said realization, and still conform to the OW procedure, by definition.

As Output Monad	As State Monad
$return\ A = \{\lvert A\rangle\ ;\ \bot\}$ $\{\lvert A\rangle\ ;\ R_i\} \gg= Op = let\ R_j = \text{‘}extract\_from\text{’}\ QOp$ $in\ let\ R = \mathbf{C\&S}\,(R_i, R_j)$ $in\ \{\lvert A\rangle\ ;\ R\}$	$return\ A = \backslash R \rightarrow (\lvert A\rangle\ ,\ R)$ $SA \gg= Op = \backslash R \rightarrow let\ (\lvert A\rangle\ ,\ R_i) = (SA\ R)$ $in\ let\ R_j = \text{‘}extract\_from\text{’}\ QOp$ $in\ let\ R = \mathbf{C\&S}\,(R_i, R_j)$ $in\ ((\lvert A\rangle)\ ,\ R)$

Figure 5. Monadic Operations for the OW model.

Now consider taking an output $\lvert B\rangle$ from Eq. (5), and apply it to another realization R_j as in

$$\lvert A\rangle \gg= \backslash(A) \rightarrow \quad Op_i\,(A) \quad \gg= \backslash\,(B) \rightarrow \tag{6}$$

$$Op_j\,(B) \quad \gg= \backslash\,(C) \rightarrow \tag{7}$$

$$return\,(C)$$

In the Step represented by Eq. (7), one must know about the previous R_i, in order to standardize its composition with R_j before applying the result onto the initial input $\lvert A\rangle$. So, in some sense, the output from the first application must incorporate both the initial state and previous realization(s) somehow.

Our definition of the monadic layer as it stands now (Fig. 3) — with *return* $A = \lvert A\rangle$, cannot accomplish that. One must at the very least lift values – say $\mathbf{A}$ – further to $\{\lvert\mathbf{A}\rangle\ ;\ \mathbf{R}\}$; with R representing the value of the C&S-ed realizations up to the current position.

As a result, by the definition of $\gg=$, at a Step such as (7) of a composition, the corresponding QOp_j – before standardization, will return $\{\lvert\mathbf{C}\rangle\ ;\ \mathbf{R_j}\}$.

Then, say we have a function ‘*extract_from*’ that takes in QOp_j and returns R_j. We define the monadic layer in Fig. 5.

As it turns out, in Programming Languages, the behavior described here corresponds to some special types of monads whose roles center around maintaining and manipulating global data, e.g., the *Output* and *State* monads.

To keep things simple, at each monadic step, we do not update the input state, but rather simply pass it along; leaving it up to later considerations to apply the associated C&S realization onto it.

Importantly, note that for all the different OW approaches we reviewed earlier in Section 2.2, the format for the monadic abstraction looks exactly the same. The only difference is in the representations of the realizations and associated C&S procedure.

However, we do have to wonder whether one could improve definitions of monadic layers somehow; and if so, what factors the process could involve. In the upcoming Section, we consider improving our OW monadic abstraction defined above. The process unveils a set of conditions for abstraction of the C&S procedure, which comes with applications in defining additional approaches to reasoning about circuits in the OW models.

For each interleaving qubit q,

(i) *A X-correction always introduces* shifts *on the neighbors (within circuit 2) of the interleaving qubit. Meanwhile, the measurement*

  (a) *either simply remains unchanged, if it is in the X-basis,*
  (b) *or remains unchanged AND introduces a* shift *on q, if it is in the Z-basis,*
  (c) **(C:)** *or becomes dependent on the correction's signals, otherwise.*

(ii) *A Z-correction always leaves the measurement unchanged AND introduces a* shift *on q.*

Figure 6. Basic Conditions for Abstraction.

3.3 Improving Abstractions

Two considerations are important, especially for translating circuits. First, we want to reduce the "context" that is being manipulated as a monad. Second, we want to improve the implementation of $\gg=$.

3.3.1 The C&S conditions

By observing the effects of standardization passes on the final pattern, as the basic C&S procedure (Alg. 1) is executed, we are able to discard those input (from subsequent applications) and output (from previous applications) qubits that are not outputs-turning-inputs, or *interleaving* qubits as we call them, from consideration.

Further, when realizations are considered to be simplified prior to executions of the C&S procedure, we also discard previous entanglements and measurements, as they no longer have any effect on subsequent applications (cf. $\sim$-decorations in Alg. 1).

Essentially, we come down to the set of conditions in Fig. 6 as a standard that any C&S procedure must meet.

Following the basic procedure sans Step 3, the conditions inform one of where *shifts* may be introduced so one can consider either propagating them — executing the optional Step 3 — for each q that is measured, or leaving them as z-dependencies[8] otherwise. Herein, the *introduction of a* shift refers to that of a z-correction that eventually translates into a z-dependency on the associated measurement (Pass v), which is then shifted out of the measurement pattern (Pass viii). Note that, due to the latter, our reasoning is biased towards patterns being simplified as in resulting from Step 3, and we use the term "dependent" to mean "x-dependent" (cf. Pass vi).

In any event, based on the C&S conditions, we abstract and improve our C&S procedures even more. Alg. 3 from the Appendix (B) describes the abstracted C&S procedure, along side a discussion of its workings.

The time complexity is linear in the number of sub-circuits, and quadratic in the input size s, with or without accounting for Step 3 of the basic C&S procedure.

[8] By Z-dependencies, we mean the effect of propagating Z-corrections over measurements (Pass v).

As Output Monad	As State Monad
$return\ A\ =\ \{\vert A\rangle\ ;\ \bot\}$ $\{\vert A\rangle\ ;\ C_i\} \gg\!= Op\ =\ let\ R_j = \text{`}extract\_from\text{'}\ QOp$ $in\ let\ (C \cdot ME) = \mathbf{C\&S}\,(C_i, R_j)$ $in\ \{ME\,(\vert A\rangle)\ ;\ C\}$	$return\ A\ =\ \backslash R\ \rightarrow\ (\vert A\rangle\ ,\ R)$ $SA \gg\!= Op\ =\ \backslash R\ \rightarrow\ let\ (\vert A\rangle\ ,\ C_i) = (SA\ R)$ $in\ let\ R_j = \text{`}extract\_from\text{'}\ QOp$ $in\ let\ (C \cdot ME) = \mathbf{C\&S}\,(C_i, R_j)$ $in\ (ME\,(\vert A\rangle)\ ,\ C)$

Figure 7. Simplified Monadic Operations for the OW model.

3.3.2 The simplified monad

With respect to our earlier monadic abstraction, we redefine our *return* as lifting $\mathbf{A}$ to $\{|\mathbf{A}\rangle\ ;\ \mathbf{C}\}$; while $\gg\!=$, on input $\{|\mathbf{A}\rangle\ ;\ \mathbf{C_i}\}$, now follows the abstract C&S procedure to combine C_i and R_j. Fig. 7 illustrates the new monad.

Note that, now, we can consistently apply the entanglement and measurement parts of patterns onto the input state, and then pass on the updated values; only leaving the application of the final sequence of corrections up for later consideration (rather than the whole realization as in previously).

Other applications of the C&S conditions introduce alternative approaches to the OW model, either extending or reducing from the abstracted C&S procedure. For example, we propose an approach that is based on a particular graphical representation for OW realizations, one that would allow one to consistently approximate circuits. We later on use this approach to illustrate our circuits translations from the SN model (Section 5).

4 A Graphical One-Way Approach

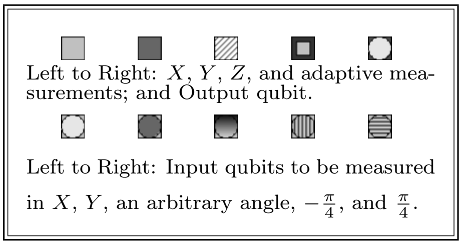
Definitions:

We start with a representation for different types of qubits, e.g., ancillae, logical inputs/outputs as in Fig. 8

Two entangled qubits are connected by a "wire" (simple line) as in □—○. Whenever possible, reduce the usage of "wires" by gluing the two qubits side by side: □○.

In a broader view, the prospect of this exercise is for the final realization to visually indicate how close it is to being a cluster state.

In all, this representation is an adaptation from Raussendorf *et al.*'s 2-dimensional lattice grid design [11], extended with a finer characterization for qubits, as well as the following.


Left to Right: X, Y, Z, and adaptive measurements; and Output qubit.

Left to Right: Input qubits to be measured in X, Y, an arbitrary angle, $-\frac{\pi}{4}$, and $\frac{\pi}{4}$.

Figure 8. Some Qubits.

To combine two circuits 1 and 2, introduce directed arrows (→) to indicate which output qubit from circuit 1 is becoming which input qubit from circuit 2.

```
Algorithm 2 A Reduced Graphical C&S Procedure.
For each interleaving qubit $q \in (O_1 \cap I_2)$:
  Consider the only measurement $m_q$ and the corrections $c_q$.
  (i) If one of the corrections satisfy $(c_q == X_q^r)$ and $(m_q == {}^t[M_q^\alpha]^{\mathbf{0}})$ :
      • Set the measurement adaptive. ______________ $m_q \leftarrow {}^t[M_q^\alpha]^{\mathbf{r}}$
  (ii) Else :
      • Leave the measurement type alone, BUT convert the representation of $q$ from input/output mode
        to ancilla.
Only adaptive measurements, and measurements in $X$, $Y$, and $\pm\frac{\pi}{4}$, are considered.
```

The standardization procedure then reduces the entire circuit representation by consistently eliminating the occurrences of arrows; while respecting the abstraction conditions in Fig. 6.

In any event, as we will see in the upcoming practical example (Section 5), an important use for this kind of visual representation for graph states, which can very quickly grow complex, is to get an approximation of the result of their combination when one can afford to overlook certain defining details. For example, let us look move onto the next Section.

4.1 Approximating Combinations: A Reduced Scheme

Whether one is missing information about, or simply less interested in, measurements outside of a restricted scope or the exact Pauli corrections, the following framework will always derive the entanglement and basic measurement patterns of a final circuit.

Essentially, and similarly to what one might find in common papers – e.g. Raussendorf *et al.*'s [11], we restrict our attention to whether measurements are adaptive and to the particular measurements in X, Y, and $\pm\frac{\pi}{4}$ when they are not. As a result, based on the abstraction conditions, interleaving qubits with adaptive (i.e. dependent) measurements need no further analysis.

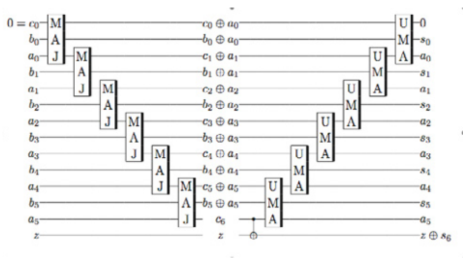
In fact, our whole concatenation procedure boils down to the procedure in Algorithm 2. Step i checks whether the condition labeled **(C:)** is satisfied, if the measurement is not already adaptive. The time complexity is a linear gain in the input size s – over the abstract procedure, and a linear gain in the number n of sub-circuits – over the BP approach.

Shall one decide to proceed ahead with Step 3 of the basic C&S procedure (Fig. 1), one can simply assume that signals accumulate in increasing order and thus never cancel out. In addition, when information about the exact corrections on an interleaving qubit is missing, it suffices to assume that it is at least X-corrected.

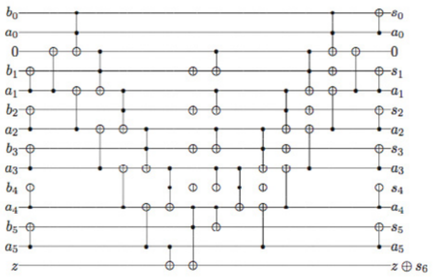
For illustration, we construct a representation for a CCZ in Fig. 9, from its abstracted representation in Fig. 4.

4.2 Generalizing the Scheme

The graphical approach defined herein is equivalent to the MC, since it is ultimately based on its patterns and their composition. The explanation lays in the following.



(a)



(b)

Figure 10. 6-qubit addition circuit (a), and its higher-parallelism version (b). [Pictures from [2]]

As one extends the reduced graphical representation (Section 4.1) to include more specific information about the measurement angles, types of corrections , e.g., *X* or *Z*, or existing signals, one moves into a more general representational framework. Conceptually, one extends the representational reasoning accordingly, modifying the associated concatenation procedure to eventually mimic the MC's completely.

5 Practical Example: A (new) One-Way Quantum Ripple Carry Addition Circuit

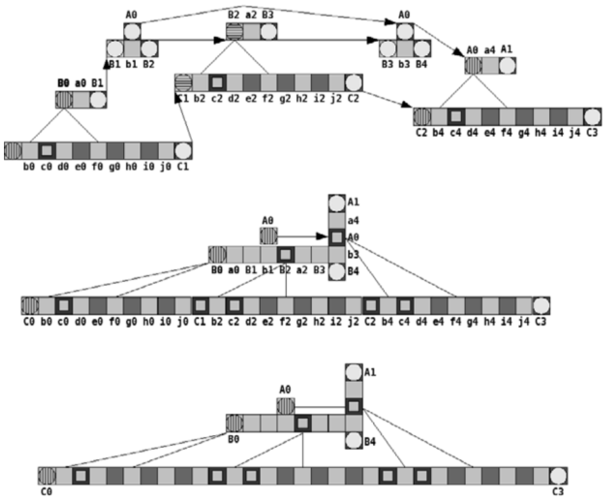
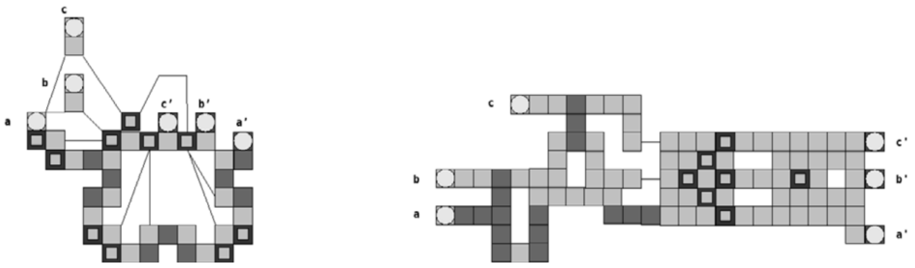


Figure 9. Building a *CCZ* gate from controlled rotations.

Consider translating the versions of Cuccaro *et al.*'s "new quantum ripple-carry addition circuit" [2] in Fig. 10. Both circuits reduce the number of ancillae needed in typical ripple-carry addition circuits (cf. [13,6,5]) down to a single one from a linear bound. If we don't count negations, they also lower the number of gates needed, relatively by the size of the input. Then, the second circuit optimizes the first one by lowering the depth, and hence producing a version with higher parallelism. Essentially, the second version re-arranges the gates in the circuit representation, compacting them to fit as many as possible within a given time-frame, after replacing the *UMA* gate with an appropriate version.

Following a building-block approach, we start our combination procedure from



(a) From general graph-states (b) From cluster-states

Figure 11. Building the MAJ gate.

deriving realizations for the MAJ and UMA (both versions [9]) subdivisions, using those for some elementary CX and CCX — or H, CZ and CCZ .

For a general graph-state realization, we can use the CCZ-realization we generated in Fig. 9. However, representations built on top of this quickly become indiscernible. For instance, just take a look at the realization of a MAJ circuit in Fig. (11a).

Luckily, we can get an alternative CCZ-realization from Raussendorf *et al.* [11]. However, this comes without explicit information about the angle of adaptive measurements, nor the signals, nor corrections in use.

We are thus left with having to apply our reduced graphical concatenation procedure. The MAJ-realization obtained here (Fig. 11b) is not exactly a cluster-state, but it is close. We eventually build up the 6-qubit addition translations in Fig. 12 from the following [10].
First, we form 1-qubit addition translations from each MAJ-UMA combination. Then, following the structure of the circuit in the SN model, we link logical inputs and outputs appropriately, using double-Hadamard transformations [11] whenever appropriate. The latter allows to (1) keep a sense of legibility from a visual perspective, and (2) maintain the cluster-like structure of the realizations.

Analysis: Optimization(s) Lost in Translation

Even though we are missing some information from the underlying CCZ-realization, we can still approximately assess the complexity difference between the two versions of the circuit in the OW model. First, we assume that all parts that correspond to MAJ or UMA subdivisions can run in parallel. Then, we take each occurrence of adaptive measurement on the underlying CCZ-realization as to be executed in a separate round of measurements. It turns out that whether we translate the original addition circuit (Fig. 10a), or its higher-parallelism version (Fig. 10b), the resulting OW realizations both have a low constant bounded depth and the difference in extra ancillae is negligible.

This does not seem to be a very surprising result. In fact, as Raussendorf *et al.*

[9] Note that in our translation of the higher-parallelism version of the circuit, we do not need to re-arrange the gates as well since that would not change the number of elementary gates we are working with.

[10] Certainly, this process is extensible to $n >$ 6-qubit addition translations.

[11] Essentially composing with the identity several times.

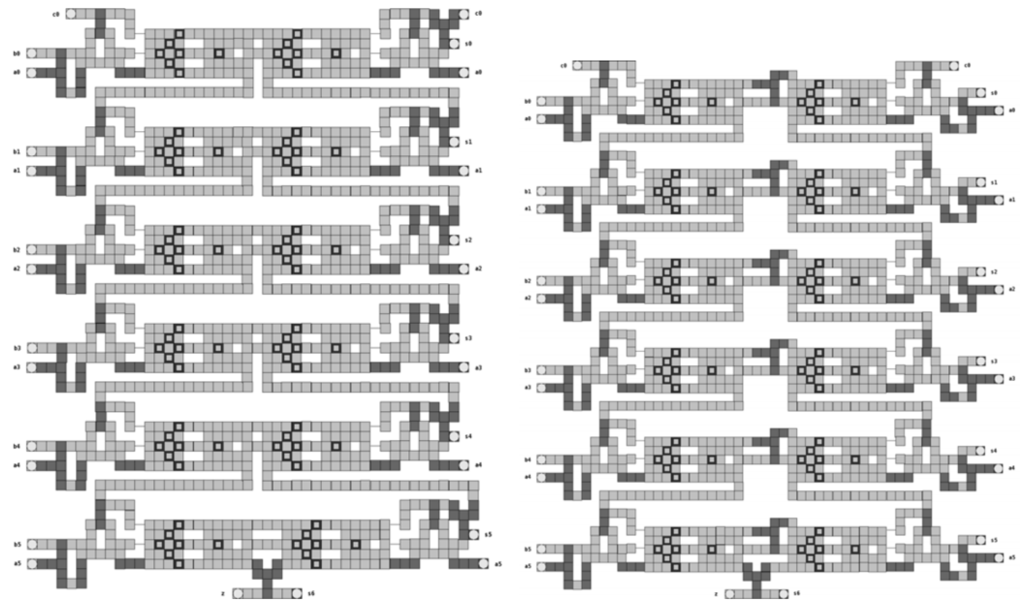

(a) (b) Higher-parallelism version

Figure 12. One-way realizations of Fig.10.

[11] points out, optimizations that manipulate gates in the Clifford group can only affect the spacial and operational resources while those involving changes in non-Clifford operations affect the final depth. Further, Danos *et al.*'s "no dependency theorems" state that operations in the Clifford group can be associated with patterns in which measurements are only in either the X or Y bases [3]; which can all be performed in a single round of execution. Therefore,

Circuit optimizations performed under the circuit model cannot be expected to make a considerable difference under the one-way model, unless they at least involve changes in non-Clifford operations.

For our example herein, the only non-Clifford effect we can be concerned with comes from the CCZ, as their decomposition in terms of controlled rotations includes $C(\pm\frac{\pi}{2})Phase$ — and $C(\pm\frac{\pi}{2})Phase$ are not in the Clifford group.

Comparing the complexity of OW realizations against the SN ones, there is not a question that the translation takes a linear procedure down to a constant one.

The only other consideration is the physical limitations on applying the OW realization, which may have one split the resulting graph into several subdivisions to be applied consecutively.

6 Conclusion: Towards a Complete Translation Framework

We considered the OW composition stage of translating circuits from the standard circuit (or network) model to the one-way one. With focus on improving the efficiency and systematization of the translation, we considered two main approaches from current literature, and abstracted them using the mathematical concept of monads; more generally, proposing an abstract model of quantum computation.

The abstraction allowed us to define some basic conditions for composing circuits in the one-way model, independently of any approach. These conditions later on served as a basis for improving the abstractions in the one-way model, including introducing an additional approach based on a graphical representation of circuits. This approach addressed the occasional need for circuits approximation and was thus perfect for the translations in our practical example, where gathered information were incomplete.

Notable here is that this idea of abstracting using monads can extend to other models, beyond the standard and the one-way models focused on here. Further, given recent developments on a generalization of monads called *arrows* [8,14], we find it potentially helpful to consider using those constructions as well.

In any event, another consideration that may be worthwhile in the long term is to define the graphical approach more formally, clearly separating the reduced monadic layers, from more general ones; perhaps modeling the additional(completing) information as monadic (or arrow-based) effects.

On an efficiency and effectiveness note, one could add optimization passes to the translation procedure, and consider the different ways that they play out in the overall complexity, with respect to the class of practical/realization needs they meet.

Last but not the least, one could also consider comparing the complexity of this compositional approach to translation against alternative ones such as the phase-map decomposition [4].

It is our hope that all these considerations would potentially lead us to a complete translation framework, that efficiently and systematically generating circuit realizations with immediate practical use.

Acknowledgement

We would like to thank all direct and indirect contributors to this work. We appreciated fruitful inputs from the Quantum Information Unit group at the Institute for Scientific Interchange, as well as the Programming Languages group at Indiana University. We would especially like to thank Profs Amr Sabry and Gerardo Ortiz for their continued guidance throughout this work.

References

[1] Browne, D. E. and H. J. Briegel, *One-way quantum computation - a tutorial introduction* (2006). URL http://www.citebase.org/abstract?id=oai:arXiv.org:quant-ph/0603226

[2] Cuccaro, S. A., T. G. Draper, S. A. Kutin and D. P. Moulton, *A new quantum ripple-carry addition circuit* (2004). URL http://www.citebase.org/abstract?id=oai:arXiv.org:quant-ph/0410184

[3] Danos, V., E. Kashefi and P. Panangaden, *The measurement calculus* (2004). URL http://www.citebase.org/abstract?id=oai:arXiv.org:quant-ph/0412135

[4] de Beaudrap, N., V. Danos and E. Kashefi, *Phase map decompositions for unitaries* (2006).
URL http://www.citebase.org/abstract?id=oai:arXiv.org:quant-ph/0603266

[5] Draper, T. G., *Addition on a quantum computer* (2000).
URL http://www.citebase.org/abstract?id=oai:arXiv.org:quant-ph/0008033

[6] Draper, T. G., S. A. Kutin, E. M. Rains and K. M. Svore, *A logarithmic-depth quantum carry-lookahead adder* (2004).
URL http://www.citebase.org/abstract?id=oai:arXiv.org:quant-ph/0406142

[7] Ekert, A., P. Hayden and H. Inamori, *Basic concepts in quantum computation* (2000).
URL http://www.citebase.org/abstract?id=oai:arXiv.org:quant-ph/0011013

[8] Hughes, J., *Generalising monads to arrows*, Science of Computer Programming **37** (1998), pp. 67–111.

[9] Mermin, N. D., "Quantum Computer Science: An introduction," Cambridge University Press, Cornell University, New York, 2007.

[10] Nielsen and Chuang, "Quantum computation and quantum information," Cambridge University Press, New York, NY, USA, 2000.

[11] Raussendorf, R., D. E. Browne and H. J. Briegel, *Measurement-based quantum computation with cluster states*, Physical Review A **68** (2003), p. 022312.
URL http://www.citebase.org/abstract?id=oai:arXiv.org:quant-ph/0301052

[12] Sabry, A., *Modeling quantum computing in Haskell*, in: *Haskell '03: Proceedings of the 2003 ACM SIGPLAN workshop on Haskell* (2003), pp. 39–49.

[13] Vedral, V., A. Barenco and A. Ekert, *Quantum networks for elementary arithmetic operations* (1995).
URL http://www.citebase.org/abstract?id=oai:arXiv.org:quant-ph/9511018

[14] Vizzotto, J., T. Altenkirch and A. Sabry, *Structuring quantum effects: superoperators as arrows*, Mathematical. Structures in Comp. Sci. **16** (2006), pp. 453–468.

A Defining the pattern for a CCZ operation

Fig. A.1 illustrates the translation from the pattern definition in Fig. 4 to that of Danos *et al.* [3], making an explicit use of the trivial pattern

$$\mathcal{T}(1) = (\,\{1\},\,\{1\},\,\{1\},\,\perp\,)\,.$$

B Abstracting the C&S procedures

Based on the abstraction conditions from Fig. 6, we modify the basic C&S procedure from Algorithm 1 into that in Algorithm 3. Note that this one primarily deals with *interleaving* qubits.

Essentially, Step 1 of the basic procedure is translated into a constant Step $1a$ here.
Steps 2 and $3a$ are translated into Steps 3 and 4; while Step 3 is now infiltrated within the new Steps 2 and 3 — precisely Steps $2a$ and $3d$, labeled **[S]**.
When the procedure is extended to concatenating n circuits, the new Step $2a$ can be performed only once, during the first round of iteration.
Both **[S]**-Steps optimize the old (basic) Step $3b$ by eliminating unnecessary traversals during *shift*-propagation steps.
In the new Steps 3 and 4, the number of measurements M_2's traversals over corrections $\tilde{C}_1$ is reduced by the number of potential extra corrections resulting from

$$
\begin{aligned}
& CCZ_{\{A0,B0,C0\}} = (\mathcal{T}_{\{B4\}} \otimes C\frac{\pi}{2}Phase_{\{A0,C2\}}) \cdot (\wedge\mathcal{X}_{\{A0,B3\}} \otimes \mathcal{T}_{\{C2\}}) \\
& \cdot(\mathcal{T}_{\{A0\}} \otimes C(-\frac{\pi}{2})Phase_{\{B2,C1\}}) \cdot (\wedge\mathcal{X}_{\{A0,B1\}} \otimes \mathcal{T}_{\{C1\}}) \cdot (\mathcal{T}_{\{A0\}} \otimes C\frac{\pi}{2}Phase_{\{B0,C0\}}) \\
= {} & Z_{C3}^{s_{i4}+s_{g4}+s_{e4}+s_{c4}+s_{C2}+s_{h4}+s_{d4}+s_{b4}+1} X_{C3}^{s_{j4}+s_{h4}+s_{f4}+s_{d4}+s_{b4}} X_{A1}^{s_{a4}} Z_{A1}^{s_{A0}+s_{e4}+s_{c4}+s_{d4}+s_{b4}} \cdot \\
& M_{a4}^{x} M_{A0}^{-\frac{\pi}{4}} M_{j4}^{x} M_{i4}^{y} M_{h4}^{x} M_{g4}^{y} M_{f4}^{x} M_{e4}^{y} M_{d4}^{x} \left[M_{c4}^{-\frac{\pi}{4}}\right]^{s_{b4}} M_{b4}^{x} M_{C2}^{-\frac{\pi}{4}} \cdot \\
& E_{a4A1} E_{A0a4} E_{j4C3} E_{i4j4} E_{h4i4} E_{g4h4} E_{f4g4} E_{A0f4} E_{e4f4} E_{d4e4} E_{c4d4} E_{b4c4}\ E_{C2b4} E_{b4A0} \\
& \cdot X_{B4}^{s_{b3}} Z_{B4}^{s_{B3}} Z_{A0}^{s_{B3}} M_{b3}^{x} M_{B3}^{x} E_{A0b3} E_{B3b3} E_{b3B4} \\
& \cdot Z_{C2}^{s_{i2}+s_{g2}+s_{e2}+s_{c2}+s_{C1}+s_{h2}+s_{d2}+s_{b2}} X_{C2}^{s_{j2}+s_{h2}+s_{f2}+s_{d2}+s_{b2}} X_{B3}^{s_{a2}} Z_{B3}^{s_{B2}+s_{e2}+s_{c2}+s_{d2}+s_{b2}+1} \cdot \\
& M_{a2}^{x} M_{B2}^{\frac{\pi}{4}} M_{j2}^{x} M_{i2}^{y} M_{h2}^{x} M_{g2}^{y} M_{f2}^{x} M_{e2}^{y} M_{d2}^{x} \left[M_{c2}^{\frac{\pi}{4}}\right]^{s_{b2}} M_{b2}^{x} M_{C1}^{\frac{\pi}{4}} \cdot \\
& E_{a2B3} E_{B2a2} E_{j2C2} E_{i2j2} E_{h2i2} E_{g2h2} E_{f2g2} E_{B2f2} E_{e2f2} E_{d2e2} E_{c2d2} E_{b2c2}\ E_{C1b2} E_{B2b2} \\
& \cdot X_{B2}^{s_{b1}} Z_{B2}^{s_{B1}} Z_{A0}^{s_{B1}} M_{b1}^{x} M_{B1}^{x} E_{A0b1} E_{B1b1} E_{b1B2} \\
& \cdot Z_{C1}^{s_{i0}+s_{g0}+s_{e0}+s_{c0}+s_{C0}+s_{h0}+s_{d0}+s_{b0}+1} X_{C1}^{s_{j0}+s_{h0}+s_{f0}+s_{d0}+s_{b0}} X_{B1}^{s_{a0}} Z_{B1}^{s_{B0}+s_{e0}+s_{c0}+s_{d0}+s_{b0}} \cdot \\
& M_{a0}^{x} M_{B0}^{-\frac{\pi}{4}} M_{j0}^{x} M_{i0}^{y} M_{h0}^{x} M_{g0}^{y} M_{f0}^{x} M_{e0}^{y} M_{d0}^{x} \left[M_{c0}^{-\frac{\pi}{4}}\right]^{s_{b0}} M_{b0}^{x} M_{C0}^{-\frac{\pi}{4}} \cdot \\
& E_{a0B1} E_{B0a0} E_{j0C1} E_{i0j0} E_{h0i0} E_{g0h0} E_{f0g0} E_{B0f0} E_{e0f0} E_{d0e0} E_{c0d0} E_{b0c0}\ E_{C0b0} E_{b0B0} \\
= {} & Z_{C3}^{\xi_{C3}} X_{C3}^{\chi_{C3}} X_{A1}^{s_{a4}} Z_{A1}^{\xi_{A1}} X_{B4}^{\chi_{B4}} Z_{B4}^{\xi_{B4}} \cdot \\
& M_{a4}^{x} M_{A0}^{-\frac{\pi}{4}} M_{j4}^{x} M_{i4}^{y} M_{h4}^{x} M_{g4}^{y} M_{f4}^{x} M_{e4}^{y} M_{d4}^{x} \left[M_{c4}^{-\frac{\pi}{4}}\right]^{s_{b4}+\mu_{C2}} M_{b4}^{x} \quad \left[M_{C2}^{-\frac{\pi}{4}}\right]^{\mu_{C2}} \cdot \\
& M_{b3}^{x} M_{B3}^{x} \cdot \\
& M_{a2}^{x} \left[M_{B2}^{\frac{\pi}{4}}\right]^{s_{b1}+s_{a0}} M_{j2}^{x} M_{i2}^{y} M_{h2}^{x} M_{g2}^{y} M_{f2}^{x} M_{e2}^{y} M_{d2}^{x} \left[M_{c2}^{\frac{\pi}{4}}\right]^{\mu_{c2}} M_{b2}^{x} \left[M_{C1}^{\frac{\pi}{4}}\right]^{\mu_{C1}} \cdot \\
& M_{b1}^{x} M_{B1}^{x} \cdot \\
& M_{a0}^{x} M_{B0}^{y} M_{j0}^{x} M_{i0}^{y} M_{h0}^{x} M_{g0}^{y} M_{f0}^{x} M_{e0}^{y} M_{d0}^{x} \left[M_{c0}^{-\frac{\pi}{4}}\right]^{s_{b0}} M_{b0}^{x} M_{C0}^{-\frac{\pi}{4}} \cdot \\
& E_{a4A1} E_{A0a4} E_{j4C3} E_{i4j4} E_{h4i4} E_{g4h4} E_{f4g4} E_{A0f4} E_{e4f4} E_{d4e4} E_{c4d4} E_{b4c4}\ E_{C2b4} E_{b4A0} \cdot \\
& E_{A0b3} E_{B3b3} E_{b3B4} \cdot \\
& E_{a2B3} E_{B2a2} E_{j2C2} E_{i2j2} E_{h2i2} E_{g2h2} E_{f2g2} E_{B2f2} E_{e2f2} E_{d2e2} E_{c2d2} E_{b2c2}\ E_{C1b2} E_{B2b2} \cdot \\
& E_{A0b1} E_{B1b1} E_{b1B2} \cdot \\
& E_{a0B1} E_{B0a0} E_{j0C1} E_{i0j0} E_{h0i0} E_{g0h0} E_{f0g0} E_{B0f0} E_{e0f0} E_{d0e0} E_{c0d0} E_{b0c0}\ E_{C0b0} E_{b0B0}
\end{aligned}
$$

with

$$
\begin{cases}
\mu_{C1} = & s_{j0} + s_{h0} + s_{f0} + s_{d0} + s_{b0} \\
\mu_{c2} = & s_{b2} + s_{b1} + \mu_{C1} + s_{a0} \\
\mu_{C2} = & s_{j2} + s_{h2} + s_{f2} + s_{d2} + s_{b2} + \mu_{C1} \\
\mu_{c4} = & s_{b4} + \mu_{c2} \\
\xi_{A1} = & s_{e4} + s_{c4} + s_{d4} + s_{b4} + s_{B3} + s_{B2} + s_{j2} + s_{h2} + s_{f2} + s_{e2} + s_{c2} + s_{b1} + s_{a0} + s_{A0} + 1 \\
\chi_{B4} = & s_{b3} + s_{a2} + s_{b1} + s_{a0} \\
\xi_{B4} = & s_{B3} + s_{B2} + s_{e2} + s_{c2} + s_{d2} + s_{b2} + s_{B1} + s_{b1} + s_{B0} + s_{j0} + s_{h0} + s_{f0} + s_{e0} + s_{c0} + s_{a0} \\
\chi_{C3} = & s_{j4} + s_{h4} + s_{f4} + s_{d4} + s_{b4} + s_{j2} + s_{h2} + s_{f2} + s_{d2} + s_{b2} + s_{j0} + s_{h0} + s_{f0} + s_{d0} + s_{b0} \\
\xi_{C3} = & s_{i4} + s_{g4} + s_{e4} + s_{c4} + s_{h4} + s_{d4} + s_{b4} + s_{C2} + s_{j2} + s_{i2} + s_{g2} + s_{f2} \\
 & +s_{e2} + s_{c2} + s_{C1} + s_{b1} + s_{C0} + s_{i0} + s_{h0} + s_{g0} + s_{e0} + s_{d0} + s_{c0} + s_{b0} + s_{a0}
\end{cases}
.
$$

Figure A.1. Working out the CCZ's pattern.

propagating the entanglements E_2 over corrections c_1 (in the old Step 1). Also, the old Step $3a$ essentially translates for free, in the same Steps as the ones introducing z-corrections.

However, these optimizations, however minor relative to big-O notation, might just cancel out with the collection of neighbors (and propagation of *shifts* through to

Algorithm 3 The C&S Procedures – Abstracted.

Standardizing $C_2M_2E_2C_1M_1E_1$ into some form $CME = \tilde{C}_2\tilde{\tilde{C}}_1\tilde{M}_2\tilde{M}_1E_2E_1$, with each $\sim$-decoration indicating a change in the pattern. $\leftarrow_p$-decorations refer to the remaining part of a pattern from point p on forward; And variables are assumed to be global.

```
1. Initialization:
   a. $E \leftarrow E_2 \cdot E_1$ ; $C \leftarrow C_2$ ; $M \leftarrow M_1$ ———————— $C_2M_2C_1M_1E_2E_1$
2. Consider $M$.
    For each $m_1$ in $M$ – from back to front: ———————— $C_2M_2\tilde{C}_1\tilde{M}_1E_2E_1$
   a. [S] reduce&prop $m_1$ $\left(C_1 \cdot \overset{\leftarrow_{m_1}}{M}\right)$
3. Consider $M_2$ and $C_1$.
    For each $m_{q_2}$ in $M_2$ – from back to front: ———————— $\tilde{C}_2\tilde{\tilde{C}}_1\tilde{M}_2\tilde{M}_1E_2E_1$
   a. $s \leftarrow 0$
   b. reduce2signal $m_{q_2}$ $s$
   c. For each $c_{q_1}$ in $C_1$ – from front to back:
     * If $q_1 == q_2$:
       (1) If $c_{q_1} == X^r_{q_1}$:
         (a) [I] $N \leftarrow$ (Collect all neighbors of $q_2$ acted upon by $\overset{\leftarrow_{m_{q_2}}}{M_2}$ )
         (b) For each $n$ in $N$:
           - [I] prop2neighbor $n$ $r$ $\overset{\leftarrow_{m_{q_2}}}{M_2}$
         (c) If $m_{q_2} == M^x_{q_2}$:
         (d) Else If $m_{q_2} == M^y_{q_2}$:
           - $s += r$ .
         (e) Else If $m_{q_2} == {}^t[M^\alpha_{q_2}]^s$:
           - $m_{q_2} \leftarrow {}^t[M^\alpha_{q_2}]^{s+r}$:
       (2) Else If $c_{q_1} == Z^r_{q_1}$:
         (a) $s += r$ .
       (3) Else:
       (4) $C_1 \leftarrow C_1 \backslash c_{q_1}$ (removing $c_{q_1}$ from $C_1$).
   d. [S] propagate $q_1$ $s$ $\left(C \cdot \overset{\leftarrow_{m_{q_2}}}{M_2}\right)$
   e. $M \leftarrow m_{q_2} \cdot M$
4. $C \leftarrow C \cdot C_1$
```

DEFINITIONS:

reduce m = Introduce a *shift* from m.
reduce&prop m *SeqList* = **reduce** m, THEN propagate the *shift* forwards across the elements in *SeqList*.
reduce2signal m sp = **reduce** m, THEN store the signal of the *shift* in sp (instead of propagating).
prop2neighbor n r *SeqList* = Propagate S^r_n forwards across *SeqList*,
until one finds the measurement ${}^t[M_n]^s$ then changes it to ${}^{t+r}[M_n]^s$.
propagate n r *SeqList* = Propagate S^r_n forwards across *SeqList*, normally.

them) in Steps $3c(1)(a)$ and $3c(1)(b)$ – labeled **[I]**.

Therefore, the time complexity for this procedure is simply that of the old Step 2. When considering Step 3, the latter result gets augmented by that of the new **[S]**-Steps. The complexity gain from the basic C&S procedure ends up being linear in the size of the input, when Step 3 is not accounted for; and linear in the number of composed circuits, when Step 3 is.