

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

Larisse D. Voufo<sup>a,c,1</sup>, Gerardo Ortiz<sup>b,2</sup> and Amr Sabry<sup>c,3</sup>

<sup>a</sup> *Institute for Scientific Interchange Foundation, 10133 Torino, Italy*

<sup>b</sup> *Department of Physics, Indiana University, Bloomington, IN 47405*

<sup>c</sup> *Department of Computer Science, Indiana University, Bloomington, IN 47405*

---

## Abstract

We present elements of quantum circuits translations from the standard network (or circuit) model to the one-way one. We present a general translation scheme, give an account of currently existing tools to apply the scheme, and propose an extension of those tools into grounds for work towards a complete translation calculus. We analyze the set of difficulties incurred from such work, and show an engendered opening to new sets of discussions and ideas. Among others, this paper extends the findings to the notions of graphical concatenation, graph state reduction (GSR) and graph state extension (GSE) passes. Further, it proposes an algorithm for running the (extended) measurement calculus with acceptable efficiency.

*Keywords:* Quantum Circuits, Models of computation, Network model, One-way model

---

## 1 Introduction

As interests and need for experiments in one-way quantum computation rise, the need for a systematic way to translate already existing circuits, especially those that derive from the standard model, into circuits obeying the one-way model, rises equivalently. In previous and related work, Schlingemann [10] systematically translates circuits from a one-way to a circuit model, but does not address the question of optimization or efficiency of the translation. Other works implicitly provide methods of circuit optimizations in a particular model through an account of possible circuit transformations and equivalence classes [9,16]. However, to the best of our knowledge, there does not exist a straightforward and systematic way to transform circuits from one given model to another, and surely not in a more-than-naive way i.e. with better (or best) efficiency and effectiveness. This poses some cumbersome

---

<sup>1</sup> Email: [voufo@isi.it](mailto:voufo@isi.it)

<sup>2</sup> Email: [ortizg@indiana.edu](mailto:ortizg@indiana.edu)

<sup>3</sup> Email: [sabry@cs.indiana.edu](mailto:sabry@cs.indiana.edu)

limits on the possibilities of implementing and testing simulations in and across arbitrary models.

This paper addresses the problem of translating a circuit from a standard network to a one-way model. We give an account of elements that such a task would normally involve, reviewing current works and extending them to more effective components such as a graphical concatenation for circuits in the one-way model. Further, we touch on the question of translations' efficiency and propose an extension of the measurement calculus [8] to include optimization passes.

As one can already derive, this paper is intended to be friendlier to the reader that is not so familiar with the field of models of quantum computation in general and with the one-way model in particular. Hence, we gather sufficient information to gain familiarity and further clarifications on current researches on the subject. In addition, we set grounds for work towards an efficient and complete translation calculus.

In the following, we assume a basic familiarity with notions of quantum mechanics and computation, fundamental differences between the circuit and the one-way models, as well as with the relation of one-way realizations to graphs.<sup>4</sup> In addition, for simplicity, we will refer to the Pauli matrices  $\sigma_x$ ,  $\sigma_y$  and  $\sigma_z$  as X, Y and Z, respectively. We will also precede any controlled operation with the letter "C".

### 1.1 A general translation scheme

In a first time, the translation can follow the structure of the circuit, decomposing it into several levels of subdivisions (in a tree-like recursive fashion) until we reach the smallest ones. The chosen one-way realizations for those smallest subdivisions will form a set that we refer to as *universal sets*. Then, at each level, associated universal sets will need to be combined accordingly based on a choice from various combination methods that we call *concatenations*. This will produce a first realization that will need to be improved based on some particular needs that we refer to as *realization needs*. We call this step *optimization*.

Alternatively, the translation can be done using the *phase map decomposition*, which bypasses completely any reference to the circuit-model representation. Instead, it analyses the possible inputs and outputs of the circuit, and generates alternative realizations for the universal set.

All combinations of these alternatives describe the possible translation paths of the circuit, which can in turn be classified in terms of the realization needs that they would meet best. Indeed, this suggests choosing the right set of unitaries and concatenation methods (plus optimizations) to start with as a detrimental factor of translation. In any events, experiments will depend heavily on the efficiency and flexibility of the associated concatenation method.

### 1.2 Related work

Raussendorf *et al.* have given a detailed account of one-way (or measurement-based) quantum computation on cluster states that provides a first universal set for clusters

---

<sup>4</sup> For more information, we refer the user to references. A tutorial version of this paper is also available for more details [17].

as well as a concatenation method through by-product operators [5]. Alternatively, Danos *et al.* have defined a more robust and parsimonious universal set [7], as well as a more standardized method of concatenation called the measurement calculus [8]. Additionally, they have also defined the phase-map decomposition [6]. Several others, such as Hein *et al.* [16] and Schlingemann [10], have studied graph state properties and their classifications into some equivalence classes that could be helpful in the process of graph state optimization.

We briefly review the concatenation methods, analysing their respective limitations, and derive an alternative scheme based on observing relationships between the graphical representations of to-be-composed one-way realizations. We call the later a *graphical concatenation*. In the process, we extend the measurement calculus to include more standardization passes as well as optimization passes derived from recent studies. We then present an example of practical interest with the translation of Cuccaro *et al.*'s quantum ripple carry addition circuit [1] and analyze the effects of the translation on the optimizations performed in the circuit model.

## 2 Building up a Translation Path: Example with the Identity operation

Let's start with the composition of two Hadamard ( $H$ ) operations into the Identity ( $I$ ). A single-wire one-way realization of  $H$  entangles a logical input qubit with a logical output one, and then measures the input in  $X$ . With respect to the randomness of measurement results, this realizes the unitary  $X_2^{s_1} H_{[1]}$  and the associated command sequence (from its pattern definition<sup>5</sup>) is  $X_2^{s_1} M_1^x E_{12}$ . The composition will have the output of the first application become the input of the second. Thus, the final realization of  $I$  will constitute of three qubits 1, 2, and 3, where qubit 2 is entangled with the remaining two and qubits 1 and 2 are measured in  $X$ .

### 2.1 Under the by-product approach [5]

The composition so described is justified by the fact that realizations meet the requirement of having their logical inputs measured in  $X$ . It can then be **verified** by composing the realized unitaries or by analysing the resulting state's characteristic eigenvalue equations; either way generating the associated by-product operators. So, the final unitary it realizes will be  $X_3^{s_2} H_{[2]} \cdot X_2^{s_1} H_{[1]}$ , and then  $X_3^{s_2} Z_{\{2\}}^{s_1} \cdot H_{[2]} \cdot H_{[1]} = X_3^{s_2} Z_3^{s_1} \cdot I_{[1]}$  after propagation of the by-product operators across the composing unitaries.

The propagation is rather **abstract**, as it deals with **unitaries** rather than automated entities, and does not guarantee the conservation of the parameters associated with unitaries that are not in the Clifford group<sup>6</sup> — so the parameters will have to be redefined accordingly. To put it bluntly, consider a non-Clifford unitary  $U(\alpha, \beta)$

<sup>5</sup> The pattern is defined according to this tuple: (*Vertices*, *LogicalInputs*, *LogicalOutputs*, *CommandSequence*).

<sup>6</sup> For example, a general rotation  $U_{rot}$  will undergo modifications according to the following relations:

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

that is equal to the product of two other unitaries  $U_1(\alpha)$  – with associated measurement pattern  $M_1(\alpha)$  and by-product operator  $C_1(\alpha)$  – and  $U_2(\beta)$  – with  $M_2(\beta)$  and  $C_2(\beta)$ .

When combining the patterns from  $U_1$  to  $U_2$ , we would like to obtain a set of measurements  $M(\alpha, \beta)$  and operators  $C(\alpha, \beta)$  for  $U(\alpha, \beta) = U_2(\beta) \cdot U_1(\alpha)$ . However, after the verification step which keeps  $M_1(\alpha)$  and  $M_2(\beta)$  constant (so that the resulting measurement pattern is  $M_2(\beta) \cdot M_1(\alpha)$ ), we get the unitary  $U_2(g(\beta)) \cdot U_1(f(\alpha))$  instead (for some functions  $f$  and  $g$ ). Thus, we go through an additional step of redefining our input parameters as  $f^{-1}(\alpha)$  (for  $\alpha$ ) and  $g^{-1}(\beta)$  (for  $\beta$ ) to fulfil our desired behavior. This then modifies the resulting measurement and correction patterns accordingly.

Thus, the task of modifying a measurement pattern for improvement, while maintaining the meaning of the realization and updating the associated by-product operator accordingly, is not trivial. Among others, in the absence of known appropriate symmetric transformations, one will have to reconsider the new input state's correlation operators, regenerating and re-evaluating its characteristic eigenvalue equations (**L1**). This is without mentioning the fact that it is practically impossible to apply this scheme to arbitrary realizations for which the realized unitary and by-product operators have yet to be determined (**L2**).

The phase-map decomposition handles the last two limitations (labeled **L1** and **L2**) and allows for trial-and-error experimentations that will uncover the realized unitary and associated by-product operator. All other limitations are addressed by the measurement calculus approach.

## 2.2 Under the measurement calculus approach [8]

Logical inputs can be measured in the **X-Y** plane of the Bloch sphere (rather than in X only). Further, the approach is more **generalized and systematic**, propagating by-product operators — now Pauli correction commands — over **measurement commands**, rather than unitaries. Thus, the earlier composition is now performed through the more automated concatenation and standardization (C&S) of patterns.

Using the symmetric transformations provided by the work of Danos *et al.* [8] and listed in Figure 1, we can standardize the composition  $X_3^{s_1} M_2^x E_{23} \cdot X_2^{s_1} M_1^x E_{12}$  by applying the sequence of passes **i**, **iii**, **iv**, and **vii** onto it. This will result in the following pattern for  $I$ :

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

In addition, we can even derive a more general realization of  $I$  by composing arbitrary  $J$  unitaries<sup>7</sup>, rather than  $H$  ones. Thus, the final pattern will be  $\mathcal{I}(1) = \mathcal{J}_\beta(2) \cdot \mathcal{J}_\alpha(1) = (\{1, 2, 3\}, \{1\}, \{3\}, X_3^{s_2} M_2^{-\beta} E_{23} \cdot X_2^{s_1} M_1^{-\alpha} E_{12})$ , and then

$$\mathcal{I}(1) = (\{1, 2, 3\}, \{1\}, \{3\}, X_3^{s_2} Z_3^{s_1} [M_2^{-\beta}]^{s_1} M_1^{-\alpha} E_{23} E_{12})$$

after running the same sequence of passes as before.

The transformations used here ensure the conservation of the meaning of realizations, while automatically updating the resulting by-product operators accordingly.

<sup>7</sup>

$$\mathcal{J}_\alpha(1) = (\{1, 2\}, \{1\}, \{2\}, X_2^{s_1} M_1^{-\alpha} E_{12}).$$

(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)	$E_{ij} A_{\mathbf{k}} \longrightarrow A_{\mathbf{k}} E_{ij}$ , 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)	$A_{\mathbf{k}} X_i^s \longrightarrow X_i^s A_{\mathbf{k}}$ , with $A \neq X$ and $A \neq Z$
(vii)	$A_{\mathbf{k}} Z_i^s \longrightarrow Z_i^s A_{\mathbf{k}}$ , 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)	$.^t [M_j^\alpha]^s S_i^r \longrightarrow S_i^r .^{t[r+s_i/s_i]} [M_j^\alpha]^{s[r+s_i/s_i]}$
(xii)	$\perp S \longrightarrow \perp$

Figure 1. Standardization passes.

---

**Algorithm 1** A simple C&S Algorithm.
 

---

(From front to back)

- (i) *Propagate each element of  $E_2$  backwards, across  $C_1$*   $\longrightarrow E = E_2.E_1$
- (ii) *Propagate each element of  $M_2$  backwards, across  $C_1$*   $\longrightarrow M = M_2.M_1$  and  $C = C_2.C_1$
- (iii) *For each  $M$* 
  - (a) *Introduce Shift*
  - (b) *Propagate the Shift forward, across the remaining part of  $M$  and  $C$ , dropping it at the end of the command sequence.*

$E_x$ ,  $M_x$ , 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 acted upon.

---

Hence, there no longer is a striking need for **verification** and non-Clifford unitaries are handled inherently. Considering the previous example with the unitary  $U(\alpha, \beta)$ , the last two and main steps of the process are no longer needed and are replaced with simply running the standardization passes. Indeed, this considerably improves the efficiency of the translation.

We propose a first intuitively efficient algorithm for the execution of those passes in Algorithm 1. The algorithm is defined for the composition of two circuits and can be applied several times for more composite circuits. It succeeds at skipping the unnecessary passes **iii** and **xii** and implicitly transfers dependencies induced by Z-actions into introduced and propagated *Shifts*.

$$\begin{aligned}
 \text{(i)} \quad & \cdot^t [M_i^{-\frac{\pi}{2}}]^s = \cdot^t [M_i^{\frac{\pi}{2}}]^{s+1} = \cdot^{t+s+1} [M_i^y] = S_i^{t+s+1} M_i^y \\
 \text{(ii)} \quad & \cdot^t [M_i^{\frac{\pi}{2}}]^s = \cdot^{t+s} [M_i^y] = S_i^{t+s} M_i^y \\
 \text{(iii)} \quad & \cdot^t [M_i^0]^s = \cdot^t [M_i^0] = S_i^t M_i^x \\
 \text{(iv)} \quad & \cdot^t [M_i^{\frac{3\pi}{4}}]^s = \cdot^{t+1} [M_i^{\frac{\pi}{4}}]^{s+1} = S_i^{t+1} [M_i^{\frac{\pi}{4}}]^{s+1} = S_i^{t+1} [M_i^{-\frac{\pi}{4}}]^s
 \end{aligned}$$

Figure 2. Extending the standardization passes.

---

**Algorithm 2** Monadic pattern for CCZ.

---

```

CCZ (A0, B0, C0) =
  C $\frac{\pi}{2}$ Phase (B0, C0) >>= \ (B1, C1) →
    CNot (A0, B1) >>= \ (A0, B2) →
  C(- $\frac{\pi}{2}$ )Phase (B2, C1) >>= \ (B3, C2) →
    CNot (A0, B3) >>= \ (A0, B4) →
  C $\frac{\pi}{2}$ Phase (B4, C2) >>= \ (B5, C3) →
  return (A0, B5, C3)

```

---

2.3 So, where is the problem?

The situation gets tedious as the size of circuits increases. Among others, patterns become more complex and less human-readable, while the C&S algorithm becomes more time-consuming. We address these problems in the following section.

### 3 Extending The Measurement Calculus

We extend the previous list of standardization passes by noticing the equalities in Figure 2. Because of their simplicity, they can be incorporated into the C&S algorithm implicitly and hence improve its efficiency further. In addition, we simplify our pattern definition so that it only pays attention to ancilla qubits as necessary, while showing a direct relationship with the corresponding network model representation.

3.1 A monadic pattern representation

Vizzotto et al. [11] earlier related “unusual” features of quantum computing (quantum parallelism and measurement) to the semantic constructions of *monads* and *arrows* from the theory of programming languages. Using their abstractions, a quantum computation expressed in the circuit model could be elegantly expressed as a computation in the monad of quantum computation.

Take, for example, the decomposition of a CCZ operation in terms of controlled rotations. Based on its circuit-model representation, we can express it as in Algorithm 2. Here, an expression  $F(i1, i2, \dots) \gg= \backslash(o1, o2, \dots)$  represents the application of some function  $F$  on some input qubits  $(i1, i2, \dots)$ , producing the outputs  $(o1, o2, \dots)$ . In addition, the arrow  $\rightarrow$  specifies the sequence of operations, while *return* specifies the final logical output qubits of the pattern.  $\gg=$  and *return* are monadic operations that hide all the quantum magic, allowing for easy translations into the previous patterns structure or a valid graphical representation.

Remarkably, both the circuit-model and the one-way realizations of the same circuit can be expressed in the exact same notation by varying the underlying monad. Figure A.1 in the Appendix shows an example of the hidden computation where the underlying monad is the pattern definition.

Alternatively, for the visual reader, the underlying monad could be graphical, representing only the entanglement relationships between the qubits as well as each one-qubit measurement<sup>8</sup>. A study of this case, not only provides further clarifications for the derivation of more complex realizations found in common one-way literatures (e.g. Raussendorf *et al.* [5]), but also brings us to an improved C&S algorithm.

### 3.2 The Graphical Concatenation

In the course of concatenating two circuits 1 and 2, with some outputs of circuit 1 ( $O_1$ ) becoming inputs of circuit 2 ( $I_2$ ) — we call them *intersecting qubits*, let's explore the standardization passes and the C&S algorithm further. For each interleaving qubit, let's examine the relationships between the measurement in  $I_2$  and the associated Pauli correction in  $O_1$ , with respect to the corresponding measurement in the resulting pattern. This yields the following conclusion, keeping in mind that each introduced *Shift* is dependant on the corrections signals of its associated qubit in  $O_1$ :

- (i) *All X-correction will always introduce Shifts on the neighbors (within circuit 2) of the intersecting qubit. Meanwhile, the measurement*
  - (a) *will either simply remain unchanged if it is in the X-observable,*
  - (b) *or remain unchanged and accompanied with a Shift, if it is in the Y-observable,*
  - (c) **(C:)** *or become dependant on the correction's signals otherwise.*
- (ii) *Meanwhile, All Z-correction on the intersecting qubit will always leave its measurement unchanged and be accompanied with a Shift.*

An alternative and possibly more efficient algorithm would stem from that in Algorithm 3, where the following definitions are used:

- $\text{reduce} \& \text{prop } m \text{ SeqList} = \text{Reduce } m$ , propagating *Shifts* forwards across the elements in *SeqList* in that order.
- $\text{prop} \& \text{neighbor } n \ r \ \text{SeqList} = \text{Propagate } S_n^r$  forwards, across *SeqList*, until we find the measurement  $.^t[M]_n^s$  and change it to  $.^{t+r}[M]_n^s$ .
- $\text{propagate } n \ r \ \text{SeqList} = \text{Propagate } S_n^r$  forwards, across *SeqList*, normally.

Compared to the previous algorithm, the *Shifts* propagation time here is still about the same, with minor optimization in Steps [A] and [C]. However, the measurement commands are traversed only once and the entanglement ones are not propagated at all. The only potential source of inefficiency is the collection of neighbors in Step [B]. Meanwhile, Step [D] is actually left optional to the user, based on the comparison of costs between removing a command from one command sequence

<sup>8</sup> In this case, the combination would be performed "by-hand" or "mentally" i.e. without the help of an implemented program.

---

**Algorithm 3** An alternative C&S Algorithm.

---

- Initialize  $E$  to  $E_2.E_1$ ,  $C$  to  $C_2$ , and  $M$  to null
  - Consider  $M_1$ . For each  $m_1$  in  $M_1$  (from front to back):
    - $M = (\text{reduce} \& \text{prop } m_1 \{ \text{remaining } M_1, C_1, M_2, \text{ and } C \}) . M$
  - Consider  $M_2$  and  $C_1$ . For each  $m_{q_2}$  in  $M_2$  (from front to back):
    - (i) **[A]** reduce  $m_{q_2}$ , saving the signal of the Shift in  $s$  (instead of propagating)
    - (ii) For each  $c_{q_1}$  in  $C_1$  (from back to front):
      - If  $c_{q_1} == q_2$ :
      - (a) If  $c_{q_1} == X_{q_1}^r$ :
        - [B]** Collect all neighbors of  $q_2$  within the remaining  $M_2$  into  $N$
        - For each  $n$  in  $N$ :
        - [C]**  $\text{prop} \& \text{neighbor } n \text{ } r \{ \text{remaining } M_2 \}$
        - If  $m_{q_2} == M_{q_2}^x$ :
        - Else If  $m_{q_2} == M_{q_2}^y$ :
        - $s + = r$  .
        - Else If  $m_{q_2} == M_{q_2}^\alpha$ :
        - $m_{q_2} = [m_{q_2}]^r$ :
        - propagate  $q_1 \text{ } s \{ \text{traversed } C_1, \text{ remaining } M_2, C \}$
      - (b) Else If  $c_{q_1} == Z_{q_1}^r$ :
        - propagate  $q_1 \text{ } s \{ \text{traversed } C_1, \text{ remaining } M_2, C \}$
      - **[D]** Move  $c_{q_1}$  to front of  $C$
    - (iii) Move  $m_{q_2}$  to back of  $M$
  - Move remaining  $C_1$  to front of  $C$
- 

and appending it to another, and repeatedly testing for equality (and ignoring if not equal); which would depend on the implementation "platform" in use. Evidently, this algorithm serves as a reliable basis for designing more efficient programming-environment-specific ones.

Another ground for improvement comes from noticing that the *Shifts*' propagations affect only those operations (measurements and corrections) that have explicit signals (different from the number 1). Thus, a design could contemplate skipping non-adaptive measurements during the *Shifts*' propagation steps as well.

### 3.2.1 A reduced graphical concatenation scheme

As patterns grow more complex, one might start being less interested in measurements outside of a restricted scope or in the exact Pauli corrections, but rather more interested in determining the entanglement and measurement patterns of the final circuit. For example, in this paper – and similarly to what one might find in common papers (such as Raussendorf *et al.* [5]), we restrict our attention to whether measurements are adaptive (i.e. dependant on previous measurements) and to the particular measurements in X, Y, and  $\pm\frac{\pi}{4}$  when they are not. Thus, intersecting qubits with adaptive measurements need no further analysis.

Indeed, this scheme can also serve in approximating the concatenation of circuits for which information about signals and Pauli corrections are missing. One simply needs to assume that intersecting qubits are X-corrected and that the *Shifts*' propa-

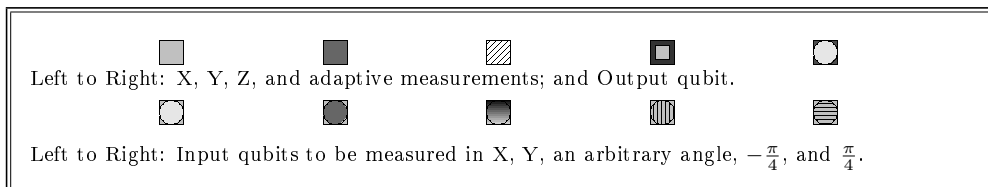


Figure 3. Graph state representation legend.

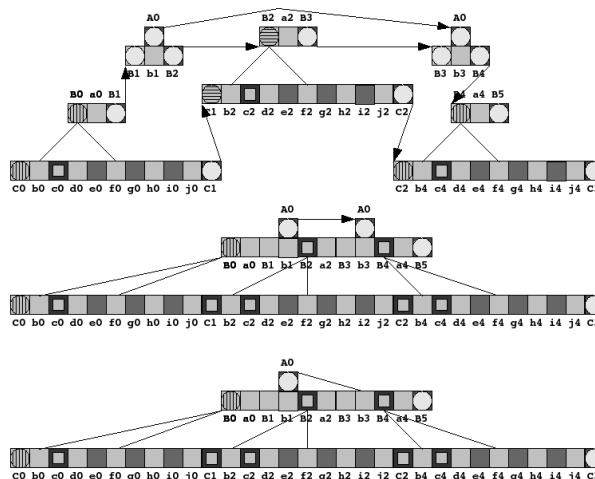


Figure 4. Building a CCZ gate from controlled rotations (graph).

gation will never cancel signals — and hence neither measurements, nor corrections.

As an illustration of the simplicity and convenience of this scheme, we derive a one-way realization of the CCZ’s pattern from Equation 1 in Figure 4. The graph is defined according to the following legend, which actually has the additional advantage of indicating approximately how close a graph state is to a cluster one.

### Graph legend

We borrow Raussendorf *et al.*’s 2-dimensional lattice grid design [5], and extend the design to include representations of logical input states that will be measured in observables different from X and Y. In relation to general graph states, we add entanglement “wires” (simple lines) for cases where maintaining the rectangular grid representation is not possible. These “wires” will hence constitute the characteristic indication that the state represented is a general graph state and not a cluster one. Finally, for step-by-step illustrations of concatenations, we also use arrows to represent which logical output qubit becomes which logical input one. Figure 3 presents the complete legend for qubits in use.

### A Reduced C&S Algorithm

At this point and under this scheme, it is easy to see that the concatenation procedure will consist of, first, determining whether the condition (C:) above is satisfied and, second, blindly converting the representation of each intersecting qubit from input- or output- qubit representations to regular-qubit (ancilla) ones, while conserving their respective derived measurement angle.

### 3.2.2 A general graphical concatenation scheme?

Further observations yield to the realization that if we extend the graphical representation to include the specification of more particular measurement angles, the type of corrections (X or Z) that are performed on each output qubit, as well as the existing signals on each measurement and correction, then we will be able to easily introduce and propagate the *Shifts* accordingly. Further, we will know the exact final pattern. But perhaps, at this point, one would find it more advantageous to program the concatenations in, defining the underlying monad as graphical representations of one-way patterns.

## 4 Optimizing the translations

Now that we have concatenated our circuits and standardized the results, it is time for some optimization. For the purpose of future work references, we classify optimizations defined in recent works into GSR and GSE transformations. We also re-express them as additional symmetric transformation passes. Indeed, we will find that each optimization pass is more appropriate for a certain set of realization needs. The particular task of arranging the order in which these optimizations are to be run, especially with respect to those desired realization needs, is similar to problems addressed by compiler optimization studies and falls outside the scope of this paper. (We are more interested in illustrating with getting a set of such passes started, based on generally known ones.) Nevertheless, we can all intuitively agree that, in order to maximize the transformations' effectiveness, by finding the smallest graph state that satisfies one's realization needs, it is preferable to run the GSR transformations prior to the GSE ones. Notice that, the execution of this step will require that measurements be allowed to be in Z, in addition to the previous limitation in the X-Y plane.

### 4.1 Graph State Reduction (GSR) transformations

As the name indicates, these concern realizations that need to use as few qubits as possible, due to either restrictions on physical space, or needs to maintain coherence.

- The removal of redundant qubits Raussendorf *et al.* [5] find that situations where a given input cluster state has far more qubits than needed need not always result in a total reconstruction of the state. Instead, it suffices to remove the extra, i.e. redundant, qubits by simply measuring them in the Z-observable. The remaining qubits will have to be modified by a phase, based on each measurement outcome and prior to any other operation. Thus, we derive the following pass to express the removal of a qubit  $q$ , from a cluster  $Q_N + \{q\}$  of  $N + 1$  qubits.

$$C_{Q_N} M_{Q_N} E_{Q_N + \{q\}} \longrightarrow C_{Q_N} M_{Q_N} Z_{Q_N}^{s_q} M_q^z E_{Q_N + \{q\}} .$$

Running the produced pattern will result in  $C_{Q_N} M_{Q_N} E_{Q_N}$  (with  $q$  tensored with the highly entangled state  $Q_N$ ) as expected. In addition, one will have to verify that  $q$  is not measured in  $M_{Q_N}$  prior to applying the pass, and also standardize the pattern after running it.

- The removal of unnecessary measurements Again, Raussendorf *et al.* [5] illustrate that whenever pairs of adjacent qubits with measurements in the X-observable are “surrounded above and below by either vacant lattices or Z-measurements, they can

be removed from the pattern without changing the logical operation of the gate” — Let’s call this “*Property X*.” In other words, consider for example a chain of four qubits  $a, b, c,$  and  $d$  such that  $a$  is entangled (CZ) with  $b,$   $b$  with  $a$  and  $c,$   $c$  with  $b$  and  $d,$  and  $d$  with  $c.$  Measuring both  $b$  and  $c$  in the  $X$ -observable is as good as not having them at all, but rather having a system consisted of only  $a$  and  $d$  entangled with one another. We derive the following as the transformation pass for removing qubits  $q_1$  and  $q_2$  from the cluster state  $Q_{N1} + Q_{N2} + \{q_1, q_2\}.$

$$C_{Q_{N1}+Q_{N2}+\{q_1,q_2\}} M_{Q_{N2}} M_{q_2}^x M_{q_1}^x M_{Q_{N1}} E_{Q_{N1}+Q_{N2}+\{q_1,q_2\}} \longrightarrow C_{Q_{N1}+Q_{N2}} M_{Q_{N2}} M_{Q_{N1}} E_{Q_{N1}+Q_{N2}} .$$

After running the pass, one will have to also remove all reference (e.g. entanglement edges and measurement outcomes) to  $q_1$  or  $q_2.$

## 4.2 Graph State Extension (GSR) transformations

Among realization needs satisfied by these transformations, are the preferences for either cluster or two-colorable states. In any event, the particular passes in the GSE case can be generalized as converses of GSR transformations.

- The introduction of  $Z$  measurements

$$C_{Q_N} M_{Q_N} E_{Q_N} \longrightarrow C_{Q_N} M_{Q_N} Z_{Q_N}^{s_q} M_q^z E_{Q_n+\{q\}} .$$

This differs from its "converse" in that  $q$  is not entangled with any other qubit in the input  $Q_N.$  Indeed, one will have to add the appropriate edges to  $E_{Q_N},$  and standardize the produced pattern after running the pass.

- The identity propagation This is simply a double-Hadamard transformation and corresponds to this pass:

$$C_{Q_{N1}+Q_{N2}} M_{Q_{N2}} M_{Q_{N1}} E_{Q_{N1}+Q_{N2}} \longrightarrow C_{Q_{N1}+Q_{N2}} M_{Q_{N2}} X_3^{s_2} Z_3^{s_1} M_{q_2}^x M_{q_1}^x M_{Q_{N1}} E_Q$$

(Assuming that  $q_2$  is entangled with  $q_3.$ ) Evidently, one will need to add edges according to “*property X*”. For example, if inserting the qubits between  $q_a$  and  $q_b,$  then the edges  $(q_a, q_1), (q_1, q_2),$  and  $(q_2, q_b)$  will have to be added.

Let us, again, clarify that these are not the only existing transformations. In addition to the Hadamard transformation itself, Schlingemann studies the removal of the Clifford parts of a given graph state in depth, and presents additional transformations [9].

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

To illustrate the applicability of our optimization passes, let’s take a look at Cuccaro *et al.*’s “new quantum ripple-carry addition circuit” [1] in Figure 5. Not only does it reduce the number of ancillae normally needed in previous ripple-carry addition circuits (cf. [12,13,14]) down to a single one from a linear bound, it also lowers the depth and the number of gates needed, producing a version with higher parallelism. Following our graphical translation scheme, we obtain the cluster-state realizations in Figures 7. In particular, the translation of the “MAJ” and “UMA” sub-circuits take advantage of the "introduction of  $Z$  measurements" feature to maintain the cluster state representation, as illustrated in Figure 6. Then, we use the "identity



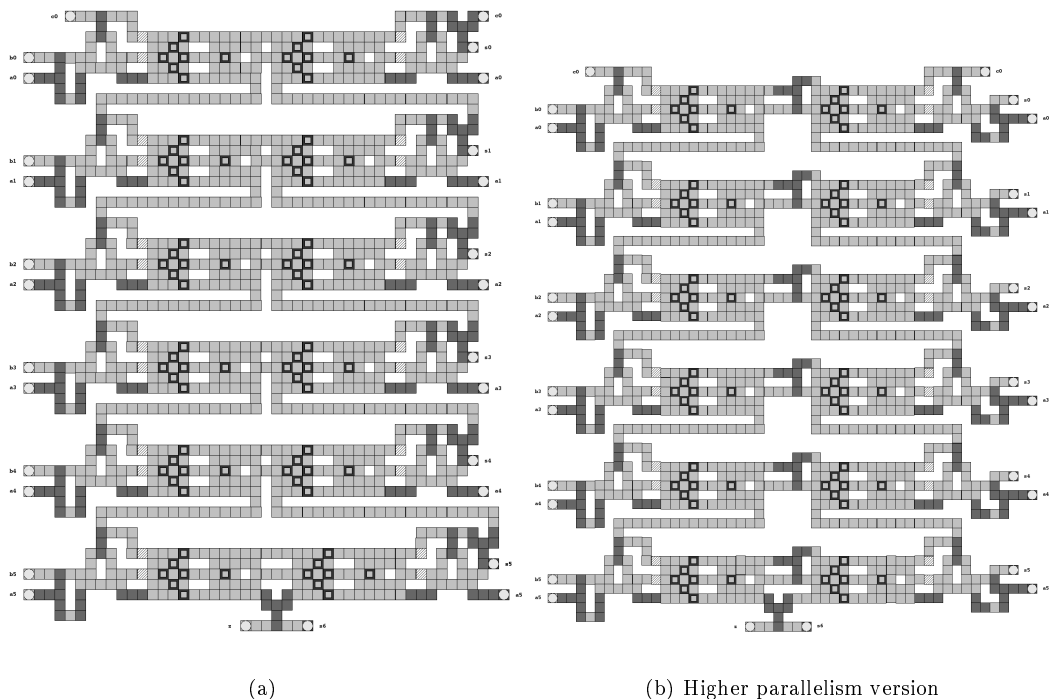


Figure 7. One-way realizations of Fig. 5 in clusters.

results? More importantly, how would the resulting translations fit with our desired realization needs? And how effective would a given translation path be with respect to the fulfilled realization needs? In other words, which translation path is more (or most) appropriate for which realization need? Could we generalize the findings into interesting theories? Furthermore, could we design and implement an efficient and effective complete translation calculus?

These are all interesting questions that we attempted to answer by giving grounds to work towards such a calculus. The later would not only extend the measurement calculus (and optimization) to measurement angles outside the X-Y plane of the Bloch sphere, but would certainly broaden our experimental range; allowing for flexible navigations between different sub-categories of graph states (e.g. clusters, 2-colorable, etc...). In fact, it could even be extended to allow navigations and analysis across different models of computation. Simply consider the inherent information hiding of our monadic pattern definition in Section 3.1. Here, we simply propose a handful of interesting discussions that would be handled better once the complete translation calculus is defined and analysed:

- Which combination of methods is best?

The experimental flexibility gained from such a calculus could result in a new class of theorems as well as in simplified paradigms of translation, respectfully of some one-way realizations' implementation needs.

- Are results from phase map decompositions optimal?

In fact, one of the resulting new theorems could specify combinations that produce optimal one-way realizations for specific implementation needs. The very definition

of a phase-map decomposition already suggests its production of an optimal realization in general, as one could specify a particular number of ancillae and work a way up until an acceptable solution is found. Independently of how efficient the specified algorithm is, the amount of backtracking involved in deriving the translated realization could be proven to be somewhat alarming. Determining the exact satisfactory point between the efficiency of the resulting realization and that of the applied algorithm is thus subject to future work.

- How well can one efficiently perform a systematic circuit's optimization?

Another point to notice here is the fact that the suggested translation scheme, after obtaining a concatenated circuit, attempts to better it by performing GSR optimizations first before the GSE ones. The idea is that whether one would like a 2-colorable or a cluster graph at the end, s/he would prefer the smallest possible. That can only happen if we reduce the concatenated circuit first. However, there are cases where performing the GSR+GSE algorithms would result in a graph that is either the same as the original one, or not that much better. Hence, we open the question of how much efficiency, in average or in particular cases, is gained from performing those optimization steps, and of whether that efficiency is worth the overhead of applying the corresponding algorithm.

Meanwhile, defining the calculus raises another set of discussions related to possible future experiments:

- When implementing the measurement calculus, could a graph theoretical approach be more effective?

When implementing a C&S algorithm, one may find it necessary to add sanitizing passes that renders all measurement angles positive, or renders them belonging to a particular set such as  $0, \frac{\pi}{2}, \frac{\pi}{4}$ , or any other needed reduction. Additionally, during concatenation, one may want to make the task as flexible as possible by not requiring that the output size of the first circuit be the same as the input size of the second. In this case, we may have to reinforce the requirement that tensor products are prioritized, still efficiently, over compositions, without relying on the user too much. It appears that one way to accomplish this is to view patterns concatenations as graphs, where the patterns are nodes, and the compositions are directed arcs from the first to the second circuit. Here, before the composition of two patterns, all patterns (including the first one) at the beginning of all incoming arcs to the second pattern should be tensored together, as well as all outgoing patterns (including the second one) of the second pattern's incoming patterns. Later on, the two tensored patterns containing respectively the two original patterns will be composed instead of the two individual ones. This result will then be more accurate, and will involve less implementation's messiness than in any other way.

- How effective would conversions of results from a graphical concatenation, to and from their pattern representations, be?

Once one is given a concatenated (and optimized) pattern description, s/he may want to convert it to the graphical representation and vice-versa, for illustration and visualization purposes. Exactly what tools are needed to accomplish this is yet another open question.

- How open would the design be to trial-and-error approaches?

This is in major part taken care of by the phase-map decomposition [6]. However, doors are open for more efficient ways.

## Acknowledgements

We would like to thank all direct and indirect contributors to this work. We especially appreciated fruitful inputs from a number of people including Dirk Schlingemann and the rest of the Quantum Information Unit group at the Institute for Scientific Interchange.

## References

- [1] Cuccaro, S., T. Draper, S. Kutin, D. Moulton. 2006. *A new quantum ripple-carry addition circuit*. arxiv:quant-ph/0410184 v1 22 Oct 2004.
- [2] Browne, D., H. Briegel. 2006. *One-way Quantum Computation*. arxiv:quant-ph/0603226 v2 3 Oct 2006.
- [3] Raussendorf, R., H. Briegel, 2000. *A One-Way Quantum Computer*. DOI: 10.1103/PhysRevLett.86.5188.
- [4] Browne, D., R. Raussendorf, H. Briegel. 2000-01. *The One-Way Quantum Computer: Computation from Correlations*. [www.qunat.org/quoxic/talks/danbrowne.ppt]
- [5] Raussendorf, R., D. Browne, H. Briegel. 2006. *Measurement-based quantum computation on cluster states*. arxiv:quant-ph/0301052 v2 14 Jan 2003.
- [6] Beaudrap, N., Danos V., E. Kashefi. *Phase-map decomposition for unitaries*. arXiv:quant-ph/0603266 v1 29 Mar 2006.
- [7] Danos, V., E. Kashefi, P. Panangaden. 2005. *Parsimonious and robust representation of unitaries in the one-way model*.
- [8] Danos, V., E. Kashefi, P. Panangaden. *The measurement Calculus*. arXiv:quant-ph/0412135 v1 17 Dec 2004.
- [9] Schlingemann, D. *Cluster states, algorithms and graphs*. arXiv:quant-ph/0305170 v2 4 Aug 2003.
- [10] Schlingemann, D. *Logical network implementation for cluster states and graph codes*. arXiv:quant-ph/0202007 v1 1 Feb 2002.
- [11] Vizzotto, J. K., T. Altenkirch, A. Sabry. 2007. *Structuring quantum effects: superoperators as arrows*. arXiv:quant-ph/0501151 v1 25 Jan 2005.
- [12] Vedral, V., A. Barenco, A. Ekert. *Quantum Networks for Elementary Arithmetic Operations*. arXiv:quant-ph/9511018 v1 16 Nov 1995.
- [13] Draper, T. G., S. A. Kutin, E. M. Rains, K. M. Svore. 2006. *A Logarithmic-Depth Quantum Carry-Lookahead Adder*. arXiv:quant-ph/0406142 v1 20 Jun 2004.
- [14] Drapper, T.G. 1998. *Addition on a Quantum Computer*. arXiv:quant-ph/0008033 v1 7 Aug 2000.
- [15] Bennet, C. H. 1973. *Logical Reversibility of Computation*.
- [16] Hein, M., J. Eisert, H. J. Briegel. 2006. *Multi-party entanglement in graph states*. arxiv:quant-ph/0307130 v7 9 Aug 2005
- [17] Voufo, L., G. Ortiz, A. Sabry. 2007. *Quantum Circuits: From a Network to a One-Way Model - A tutorial Version*. Technical Report TR665. Computer Science Department - Indiana University. <http://www.cs.indiana.edu/cgi-bin/techreports/TRNNN.cgi?trnum=TR665>

$$\begin{aligned}
 CCZ_{\{A_0, B_0, C_0\}} &= (\mathcal{T}_{(A_0)} \otimes C^{\frac{\pi}{2}} \text{Phase}_{\{B_4, C_2\}}) \cdot (\wedge \mathcal{X}_{\{A_0, B_3\}} \otimes \mathcal{T}_{(C_2)}) \cdot (\mathcal{T}_{(A_0)} \otimes C(-\frac{\pi}{2}) \text{Phase}_{\{B_2, C_1\}}) \\
 &\quad \cdot (\wedge \mathcal{X}_{\{A_0, B_1\}} \otimes \mathcal{T}_{(C_1)}) \cdot (\mathcal{T}_{(A_0)} \otimes C^{\frac{\pi}{2}} \text{Phase}_{\{B_0, C_0\}}) \\
 &= Z_{C_3}^{s_{i4}+s_{g4}+s_{e4}+s_{c4}+s_{C2}+s_{h4}+s_{d4}+s_{b4}+1} X_{C_3}^{s_{j4}+s_{h4}+s_{f4}+s_{d4}+s_{b4}} X_{B_5}^{s_{a4}} Z_{B_5}^{s_{B4}+s_{e4}+s_{c4}+s_{d4}+s_{b4}} \\
 &\quad M_{a4}^x M_{B_4}^{-\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_{C_2}^{-\frac{\pi}{4}} \\
 &\quad E_{a4B5} E_{B4a4} E_{j4C3} E_{i4j4} E_{h4i4} E_{g4h4} E_{f4g4} E_{B4f4} E_{e4f4} E_{d4e4} E_{c4d4} E_{b4c4} E_{C2b4} E_{b4B4} \\
 &\quad \cdot X_{B4}^{s_{b3}} Z_{B4}^{s_{B3}} Z_{A_0}^{s_{B3}} M_{b3}^x M_{B_3}^x E_{A0b3} E_{B3b3} E_{b3B4} \cdot \\
 &\quad Z_{C_2}^{s_{i2}+s_{g2}+s_{e2}+s_{c2}+s_{C1}+s_{h2}+s_{d2}+s_{b2}} X_{C_2}^{s_{j2}+s_{h2}+s_{f2}+s_{d2}+s_{b2}} X_{B_3}^{s_{a2}} Z_{B_3}^{s_{B2}+s_{e2}+s_{c2}+s_{d2}+s_{b2}+1} \\
 &\quad M_{a2}^x M_{B_2}^{\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_{C_1}^{\frac{\pi}{4}} \\
 &\quad 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} \\
 &\quad \cdot X_{B_2}^{s_{b1}} Z_{B_2}^{s_{B1}} Z_{A_0}^{s_{B1}} M_{b1}^x M_{B_1}^x E_{A0b1} E_{B1b1} E_{b1B2} \cdot \\
 &\quad Z_{C_1}^{s_{i0}+s_{g0}+s_{e0}+s_{c0}+s_{C0}+s_{h0}+s_{d0}+s_{b0}+1} X_{C_1}^{s_{j0}+s_{h0}+s_{f0}+s_{d0}+s_{b0}} X_{B_1}^{s_{a0}} Z_{B_1}^{s_{B0}+s_{e0}+s_{c0}+s_{d0}+s_{b0}} \\
 &\quad M_{a0}^x M_{B_0}^{-\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_{C_0}^{-\frac{\pi}{4}} \\
 &\quad 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_{C_3}^{\xi_{C3}} X_{C_3}^{\chi_{C3}} X_{B_5}^{s_{a4}} Z_{B_5}^{\xi_{B5}} Z_{A_0}^{\xi_{A0}} \cdot \\
 &\quad M_{a4}^x \left[ M_{B_4}^{-\frac{\pi}{4}} \right]^{s_{b3}+s_{a2}+s_{b1}+s_{a0}} M_{j4}^x M_{i4}^y M_{h4}^x M_{g4}^y M_{f4}^x M_{e4}^y M_{d4}^x \cdot \\
 &\quad \left[ M_{c4}^{-\frac{\pi}{4}} \right]^{\mu_{c4}} M_{b4}^x \left[ M_{C_2}^{-\frac{\pi}{4}} \right]^{\mu_{C2}} M_{b3}^x M_{B_3}^x \cdot \\
 &\quad M_{a2}^x \left[ M_{B_2}^{\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 \cdot \\
 &\quad \left[ M_{c2}^{\frac{\pi}{4}} \right]^{\mu_{c2}} M_{b2}^x \left[ M_{C_1}^{\frac{\pi}{4}} \right]^{\mu_{C1}} M_{b1}^x M_{B_1}^x \cdot \\
 &\quad M_{a0}^x M_{B_0}^{-\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_{C_0}^{-\frac{\pi}{4}} \cdot \\
 &\quad E_{a4B5} E_{B4a4} E_{j4C3} E_{i4j4} E_{h4i4} E_{g4h4} E_{f4g4} E_{B4f4} E_{e4f4} E_{d4e4} E_{c4d4} E_{b4c4} E_{C2b4} E_{b4B4} \cdot \\
 &\quad E_{A0b3} E_{B3b3} E_{b3B4} \cdot \\
 &\quad 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 \\
 &\quad E_{A0b1} E_{B1b1} E_{b1B2} \cdot \\
 &\quad 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

$$\begin{cases}
 \mu_{C1} = s_{j0} + s_{h0} + s_{f0} + s_{d0} + s_{b0} \\
 \mu_{c2} = s_{b2} + s_{b1} + \mu_{C1} + s_{a0} \\
 \mu_{C2} = s_{j2} + s_{h2} + s_{f2} + s_{d2} + s_{b2} + \mu_{C1} \\
 \mu_{c4} = s_{b4} + s_{b3} + s_{j2} + s_{h2} + s_{f2} + s_{d2} + \mu_{c2} + s_{a2} = s_{b4} + s_{b3} + \mu_{C2} + s_{a2} + s_{b1} + s_{a0} \\
 \xi_{A0} = s_{B3} + s_{b3} + s_{B2} + s_{j2} + s_{h2} + s_{f2} + s_{e2} + s_{c2} + s_{a2} + 1 \\
 \xi_{B5} = s_{B4} + s_{e4} + s_{c4} + s_{d4} + s_{b4} + s_{B1} + s_{B0} + s_{e0} + s_{d0} + s_{c0} + s_{b0} + \xi_{A0} + 1 \\
 \chi_{C3} = s_{j4} + s_{h4} + s_{f4} + s_{d4} + s_{b4} + s_{j2} + s_{h2} + s_{f2} + s_{d2} + s_{j0} + s_{h0} + s_{f0} + s_{d0} + s_{b0} \\
 \xi_{C3} = s_{i4} + s_{g4} + s_{e4} + s_{c4} + s_{h4} + s_{d4} + s_{b4} + s_{b3} + s_{C2} + s_{j2} + s_{i2} + s_{g2} + s_{f2} + s_{e2} \\
 \quad + s_{c2} + s_{a2} + s_{C1} + s_{C0} + s_{i0} + s_{h0} + s_{g0} + s_{e0} + s_{d0} + s_{c0} + s_{b0}
 \end{cases}$$

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

## A Defining the pattern for a CCZ operation

Figure A.1 illustrates the translation from the pattern definition in Algorithm 2 (monadic) to that of Danos *et al.* [8], making an explicitly use of the trivial pattern

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