Graphs That Calculate

Young Quantum Researcher Summer School
3 July 2026, University of Edinburgh

In this miniproject we will learn that quantum computations can be drawn as graphs called *ZX-diagrams*. We will also ask a more mathematical question: what does it mean for a graph to represent a calculation? We will attach matrices to diagrams, study the "unambiguity theorem", and finish with a glimpse of how diagrams can be rewritten to optimise quantum circuits.

0 Homework

One of the main features of a compiler is the optimisation of a program: can I represent a program in a more efficient way whilst preserving its meaning? In quantum programming, this optimisation can be performed using the ZX-calculus, a graphical language for representing quantum processes. The ZX-calculus comes with its own set of rewrite rules which we use for optimising a diagram by reducing the number of operations used.

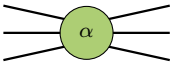
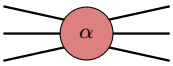
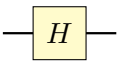
This mini-project is about learning the core specification and properties of ZX-calculus and to perform some rewrite calculations on it. As a preparation, please read this little introduction: `https://zxcalculus.com/intro.html`. You can also start to work on the first section of this worksheet.

1 Diagrams as mathematical objects

1.1 ZX-diagrams

We draw inputs on the left and outputs on the right. An (n, m)-diagram has n input wires and m output wires. For example, a $(2, 3)$-diagram has two wires coming in and three wires going out.

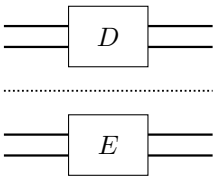
The basic pieces are as follows.

Piece	Picture	Type and syntactic meaning
Straight wire		$(1, 1)$: one input continues to one output.
Swap		$(2, 2)$: two wires exchange places.
Cup		$(0, 2)$: no input, two outputs.
Cap		$(2, 0)$: two inputs, no output.
Z-spider		(n, m): any number of inputs and outputs, with phase α.
X-spider		(n, m): any number of inputs and outputs, with phase α.
Hadamard box		$(1, 1)$: the one-qubit Hadamard operation.

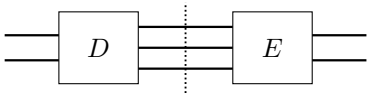
When the phase is 0, we usually leave it off the spider.

Two ways to combine diagrams.

- *Parallel composition*: put one diagram above the other. The new diagram has all the inputs and outputs from both diagrams.
- *Sequential composition*: connect the output wires of the first diagram to the input wires of the second, from left to right.



parallel composition
one above the other



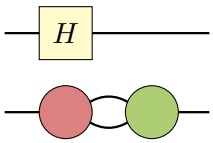
sequential composition
connected left to right

Exercise 1 (warm-up drawings)**.** Draw four different $(2, 1)$-diagrams.

1.2 Ambiguity.

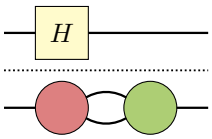
When we draw a ZX-diagram, we see the finished picture. But that picture may have been made in more than one way. We can build some parts first and then put them together, or build different parts first and still end up with the same picture.

For example, consider this $(2, 2)$-diagram:



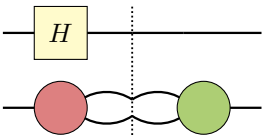
One construction builds the two rows first: the top row is a $(1, 1)$-diagram, and the bottom row is also a $(1, 1)$-diagram; then we put those two rows one above the other. A second construction builds vertical slices first: make H above X, make a straight wire above Z, and then connect the two slices from left to right.

Rows first



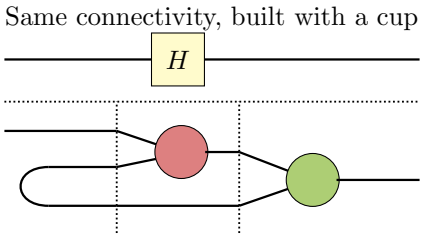
build each row,
then stack the rows

Slices first



build each slice,
then connect left to right

The same connectivity can also be built using a cup. In the lower part of the diagram below, the red spider is a $(2, 1)$-X spider and the green spider is a $(2, 1)$-Z spider. The cup has two outputs: one goes into the X-spider, and the other goes into the Z-spider. The dotted lines mark one possible order in which the pieces are assembled.



These are different construction histories, but they produce the same final connectivity. We could make even more histories by stretching wires, bending them, or inserting many straight-wire pieces before the H. The grammar is therefore *ambiguous*: the finished diagram does not remember a unique way in which it was assembled.

2 Matrices as meanings

2.1 Definition of Matrices

We now attach a matrix to a ZX-diagram. This matrix is the *meaning/semantics* of the diagram: it tells us which calculation the diagram represents.

For an (n, m)-diagram, the meaning will be a $2^m \times 2^n$ matrix U. Here is how to read it. First choose values for the n input wires. Each input wire is labelled by either $|0\rangle$ or $|1\rangle$, and we read the labels from top to bottom as a length-n bit string s, such as 010. The output is allowed to be a superposition of all length-m bit strings t:

$$\sum_{t:\text{length-}m\text{ bit string}} u_{t,s}|t\rangle.$$

The matrix U is obtained by arranging all these coefficients $u_{t,s}$. Rows are labelled by output strings t, columns are labelled by input strings s, and the (t, s)-entry is the coefficient of $|t\rangle$ when the input was $|s\rangle$.

Example: reading a matrix as a computation

Take rows and columns in the order $00, 01, 10, 11$. Rows are output strings, and columns are input strings. The matrix

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

means:

$$|00\rangle \mapsto |00\rangle, \qquad |01\rangle \mapsto |01\rangle,$$
$$|10\rangle \mapsto \frac{1}{\sqrt{2}}|10\rangle + \frac{1}{\sqrt{2}}|11\rangle = |1+\rangle,$$
$$|11\rangle \mapsto \frac{1}{\sqrt{2}}|10\rangle - \frac{1}{\sqrt{2}}|11\rangle = |1-\rangle.$$

How do we define the matrix for a diagram? We follow the way in which the diagram was built from the basic pieces. Because of ambiguity, the same finished picture may have several possible construction histories. For the moment, we treat a diagram as a picture together with a remembered construction history. The definitions below use that remembered history.

The matrices of the basic pieces.

The notation $[\![D]\!]$ means "the matrix meaning of the diagram D". When a picture is written inside the brackets, the brackets mean: take that picture and replace it by its matrix.

$$\left[\!\left[\ \text{—}\ \right]\!\right] = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad \left[\!\left[\ \cup\ \right]\!\right] = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \qquad \left[\!\left[\ \cap\ \right]\!\right] = \begin{pmatrix} 1 & 0 & 0 & 1 \end{pmatrix}.$$

The swap exchanges the two wires. With rows and columns ordered as $00, 01, 10, 11$, it sends $|ab\rangle$ to $|ba\rangle$:

$$\left[\!\left[\ \times\ \right]\!\right] = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

For the Hadamard box,

$$\left[\!\left[\ \text{—}\boxed{H}\text{—}\ \right]\!\right] = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

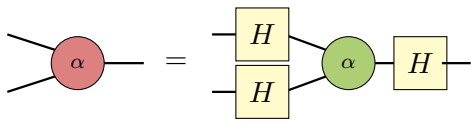
For a Z-spider with n inputs, m outputs and phase α, the rule is:

$$\left[\!\left[Z^{\alpha}_{n,m} \right]\!\right]_{t,s} = \begin{cases} 1, & \text{if every bit in } s \text{ and } t \text{ is } 0, \\ e^{i\alpha}, & \text{if every bit in } s \text{ and } t \text{ is } 1, \\ 0, & \text{otherwise.} \end{cases}$$

Here $e^{i\alpha}$ is a complex number defined as

$$e^{i\alpha} = \cos(\alpha) + i \sin(\alpha).$$

The X-spider is the same idea, but written in the $|+\rangle$, $|-\rangle$ basis rather than the $|0\rangle$, $|1\rangle$ basis. Equivalently: put a Hadamard box on every input wire, use a Z-spider, and then put a Hadamard box on every output wire. In matrix notation this is



$$\left[\!\left[X^{\alpha}_{n,m} \right]\!\right] = \left[\!\left[H \right]\!\right]^{\otimes m} \left[\!\left[Z^{\alpha}_{n,m} \right]\!\right] \left[\!\left[H \right]\!\right]^{\otimes n}.$$

The symbol $[\![H]\!]^{\otimes n}$ means: put n copies of the Hadamard matrix side by side as a tensor product.

Parallel composition becomes tensor product. Suppose D is placed above E, and call the resulting stacked diagram P. Then

$$[\![P]\!] = [\![D]\!] \otimes [\![E]\!], \qquad ([\![D]\!] \otimes [\![E]\!])_{tt',ss'} = [\![D]\!]_{t,s} [\![E]\!]_{t',s'}.$$

Here $\otimes$ denotes the tensor product of matrices, not the act of stacking diagrams. In words: multiply the entries belonging to the top diagram and the bottom diagram. Look up "Kronecker product" on Wikipedia to get a more detailed definition.

Sequential composition becomes matrix multiplication. Suppose the output wires of D are connected to the input wires of E, and call the resulting connected diagram Q. The diagram is read left to right, but the matrices act from the left, so the matrix order is $[\![E]\!][\![D]\!]$:

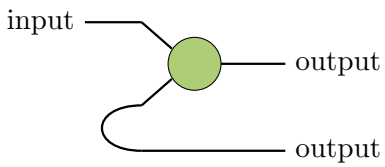
$$[\![Q]\!] = [\![E]\!][\![D]\!], \qquad [\![Q]\!]_{u,s} = \sum_t [\![E]\!]_{u,t} [\![D]\!]_{t,s}.$$

This is ordinary matrix multiplication. In words: sum over all possible bit strings on the wires that were glued together. On Wikipedia, this is explained in the article on "Matrix multiplication".

2.2 Examples and Unambiguity Theorem

Because the grammar of ZX-diagrams is ambiguous, the same finished picture can come with more than one construction history. Since our matrix meaning was defined by following a construction history, this raises a question: could different histories give different matrices? In the examples below, we calculate matrices for diagrams that are built in different ways but have the same final connectivity.

Worked example: bending a leg of a Z-spider. Now consider a Z-spider with two inputs and one output, with a cup attached to one input. The result is a diagram with one input and two outputs.



Its remembered construction history is this: first put a straight wire above a cup; then connect the three wires to a two-input, one-output Z-spider placed above a straight wire. Call the whole diagram T.

First calculate the matrix for the straight wire placed above the cup. Columns are ordered by the input bit $0, 1$, and rows are ordered as $000, 001, 010, 011, 100, 101, 110, 111$:

$$A := I \otimes \text{cup} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}.$$

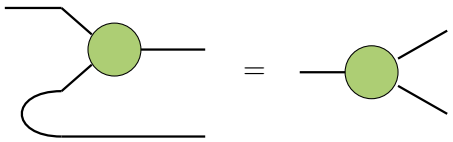
Next calculate the matrix for the $Z_{2,1}$-spider placed above a straight wire. Columns are ordered as $000, 001, 010, 011, 100, 101, 110, 111$, and rows are ordered as $00, 01, 10, 11$:

$$B := Z_{2,1} \otimes I = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

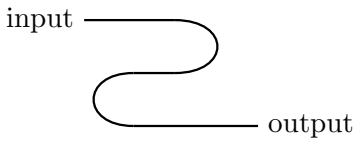
Now multiply the matrices:

$$[\![T]\!] = BA = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}.$$

This is exactly the matrix for $Z_{1,2}$. In diagrams:



Worked example: a bent wire is still a wire. The following diagram is built from one wire, one cup and one cap. It looks bent, but it should have the same meaning as a straight wire.



Its remembered construction history is this: first put a straight wire above a cup; then connect the three wires to a cap placed above a straight wire. Call the whole diagram B.

The first matrix is the same as before:

$$A := I \otimes \text{cup} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}.$$

For the next matrix, use binary order for the columns and $0, 1$ for the rows:

$$C := \text{cap} \otimes I = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

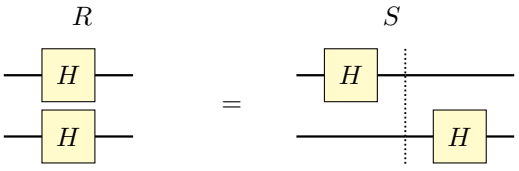
Multiplying gives

$$[\![B]\!] = CA = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Exercise 2 (Order of construction)**.** Let R be the diagram with one H on the top wire and one H on the bottom wire. Let S be the diagram built in two steps: first put H on the top wire and a straight wire below it; then connect it to a straight wire on top and H on the bottom wire. Check by matrix multiplication that

$$[\![R]\!] = [\![H]\!] \otimes [\![H]\!] \qquad \text{and} \qquad [\![S]\!] = (I \otimes [\![H]\!])([\![H]\!] \otimes I),$$

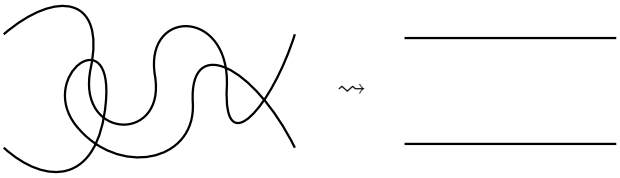
and show that these two matrices are equal.



Unambiguity Theorem We have seen that a diagram can have many construction histories. We have also seen, by calculation, that some different histories give the same matrix. The general statement is a standard fact about ZX-diagrams. We will not prove it today, but we will use it as the reason that the matrix meaning does not depend on the particular construction history we happened to choose.

Theorem (Unambiguity Theorem)**.** *If two ZX-diagrams have the same connectivity, then they represent the same matrix, no matter how they were built from the basic pieces.* □

This is why it is safe to draw diagrams flexibly. Stretching a wire, bending it round, or decomposing a picture into pieces in a different order does not change the calculation.

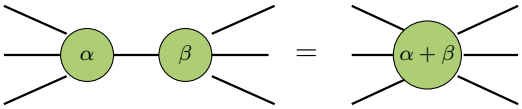


3 Rewriting diagrams

ZX-calculus is a collection of rules for replacing one diagram by another diagram with the same matrix meaning. These rules let us simplify graphs while preserving the calculation.

Thanks to the unambiguity theorem, we are allowed to treat the picture as the main object. We do not need to remember exactly how it was assembled from smaller pieces. This is what makes rewriting diagrams safe: if a rule tells us that two pictures have the same matrix meaning, then we can replace one by the other without changing the calculation.

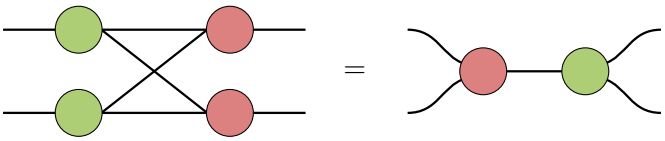
One important rule is *spider fusion*: two connected spiders of the same colour can fuse into one spider, and their phases add.



For the simple $(1, 1)$ case, this is just matrix multiplication:

$$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\alpha} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & e^{i\beta} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i(\alpha+\beta)} \end{pmatrix}.$$

Another important rule is the *bialgebra rule*. It says that two green spiders connected to two red spiders in the complete crossed pattern can be replaced by one red spider connected to one green spider, with two wires on each outside edge. This is one reason that different-looking ZX-diagrams can still represent the same matrix.



Exercise 3 (more rewrite rule)**.** Visit the Wikipedia page on ZX-calculus, or Section 3.2 of *Picturing Quantum Processes* [4], and look at the list of rewrite rules. Choose one rule that is not spider fusion, then prove the corresponding matrix equality for a small case.

4 From graphs back to quantum circuits

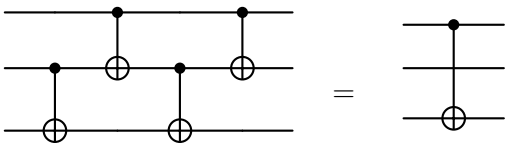
The benefit of ZX-calculus is that it turns large matrix calculations into pictures. Instead of multiplying huge matrices directly, we can look at how the diagram is connected and use a small number of rewrite rules to simplify it. This makes many calculations more intuitive: the important information is carried by the connectivity of the graph, together with the colours and phases of the spiders.

In practice, this gives the following workflow:

1. Start with a quantum circuit.
2. Translate the circuit into a ZX-diagram.
3. Rewrite the ZX-diagram into a simpler diagram, without changing its matrix.
4. Translate the simpler diagram back into a quantum circuit.

If the final circuit uses fewer gates, fewer wires, or a simpler layout, the ZX calculation has helped to *optimise* the quantum computation. Strategies like this are used in compilers for quantum programming languages.

Exercise 4 (Circuit translation)**.** Look at Section 8.2 of *Picturing Quantum Processes* [4], containing a table of quantum gates and their corresponding ZX-diagrams and matrices. Using the rules of ZX-calculus, show that the following two quantum ciruits are equivalent:



5 Further Reading

- **More about ZX-calculus.** If you want to learn ZX-calculus beyond this worksheet, start with Aleks Kissinger and John van de Wetering's book *Picturing Quantum Software: An Introduction to the ZX-Calculus and Quantum Compilation* [4]. For a university lecture-note view of how ZX-calculus fits into quantum programming, see Chris Heunen's *Introduction to Quantum Programming and Semantics*: `https://opencourse.inf.ed.ac.uk/sites/default/files/https/opencourse.inf.ed.ac.uk/iqps/2026/iqps_2.pdf`.
- **Optimisation using ZX.** ZX-calculus can be used to simplify quantum circuits: translate a circuit into a diagram, rewrite the diagram, and translate the result back into a smaller circuit. A practical starting point is the PyZX documentation: `https://pyzx.readthedocs.io/`.
- **ZX-calculus and quantum error correction.** Diagrammatic methods are useful for reasoning about quantum error-correcting codes, where the aim is to protect quantum information from mistakes. One paper that uses graphical structures to design and verify quantum error-correcting codes is *Graphical Structures for Design and Verification of Quantum Error Correction* [2].

- **Other diagrammatic calculi.** ZX-calculus is not the only graphical language for quantum computation. Two useful examples are the ZW-calculus, which is closely related to GHZ and W entanglement, and the ZH-calculus, which is useful for circuits with classical non-linear behaviour such as AND-like operations [1, 3].
- **Category theory and string diagrams.** There is a beautiful mathematical idea hiding behind these pictures. Category theory is a language for recognising the same algebraic pattern in many different places, not just in quantum computation. String diagrams are its way of turning those patterns into pictures. If the rules in this worksheet felt surprisingly natural, this is one reason why.

References

[1] Miriam Backens and Aleks Kissinger: ZH: A Complete Graphical Calculus for Quantum Computations Involving Classical Non-linearity, available at: `https://arxiv.org/abs/1805.02175`.

[2] Nicholas Chancellor, Aleks Kissinger, Joschka Roffe, Stefan Zohren, Dominic Horsman: Graphical Structures for Design and Verification of Quantum Error Correction, available at: `https://arxiv.org/abs/1611.08012`.

[3] Amar Hadzihasanovic: A Diagrammatic Axiomatisation for Qubit Entanglement, available at: `https://arxiv.org/abs/1501.07082`.

[4] Aleks Kissinger and John van de Wetering: Picturing Quantum Software: An Introduction to the ZX-Calculus and Quantum Compilation, available at: `https://github.com/zxcalc/book`.