Quantum Computing with Pictures

Stefano Gogioso

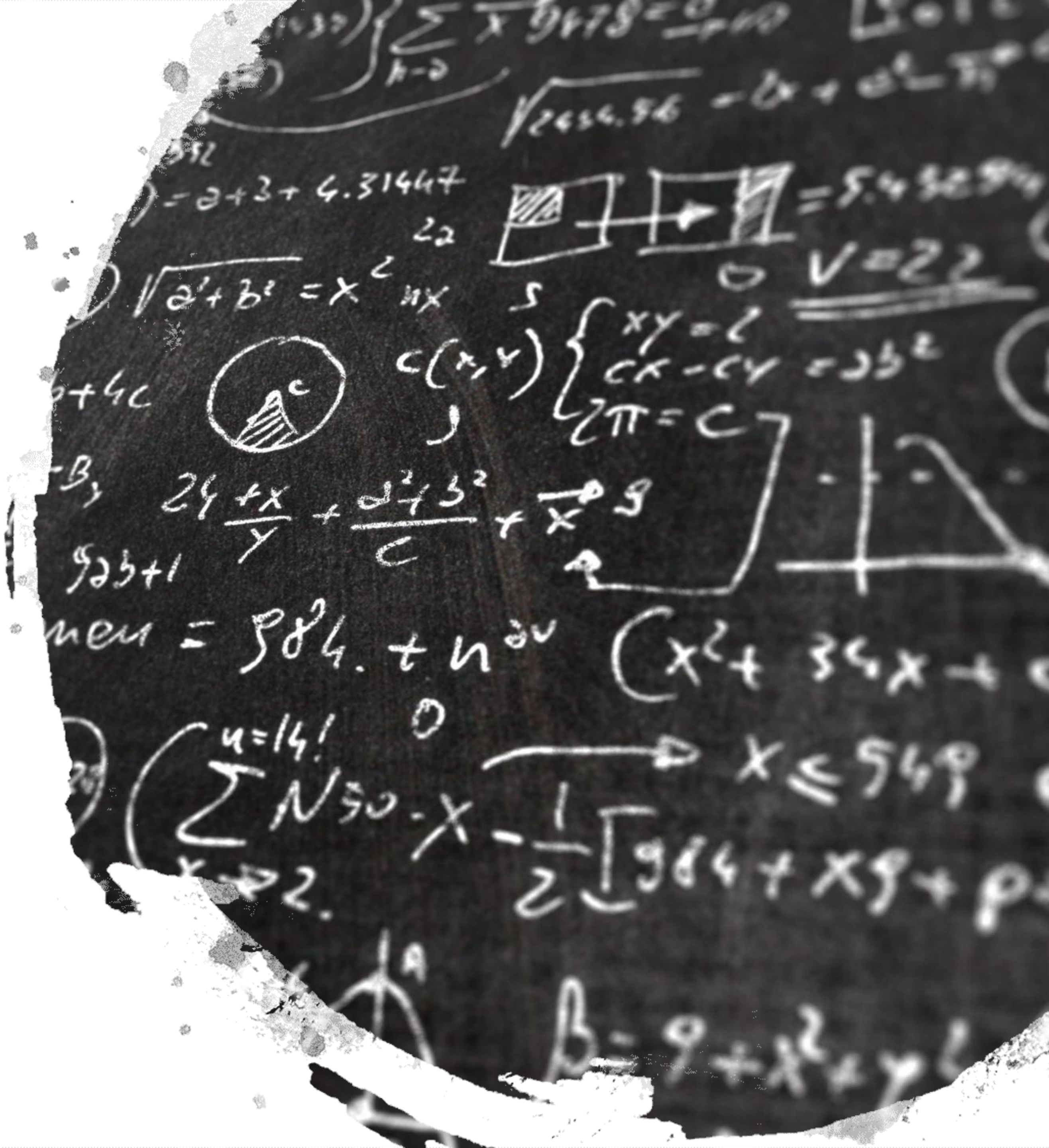
Quantum Group

University of Oxford

In this Masterclass

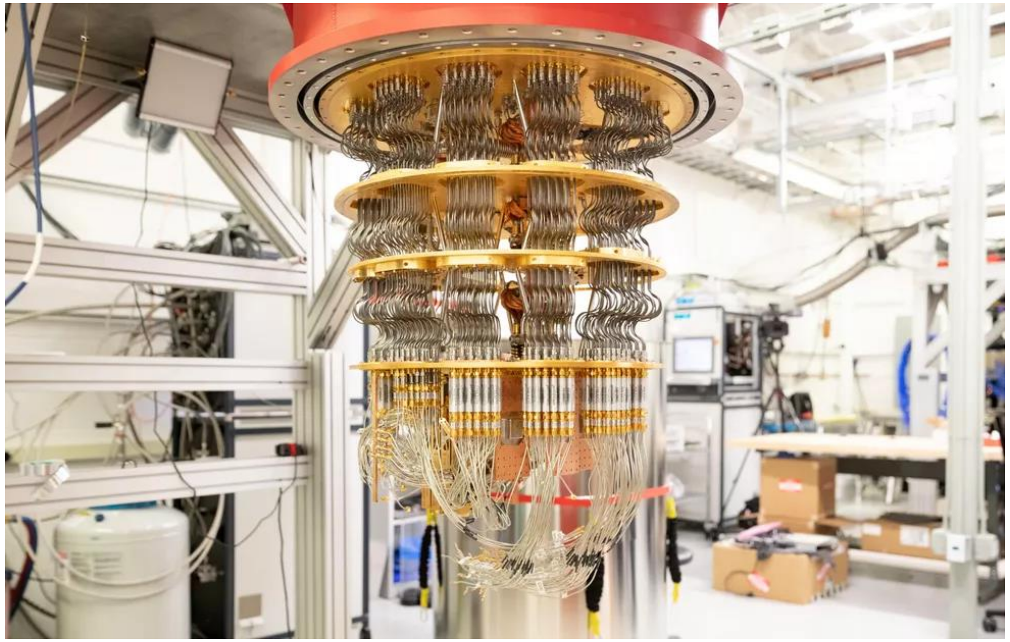
We will learn how to do the following, using pictures:

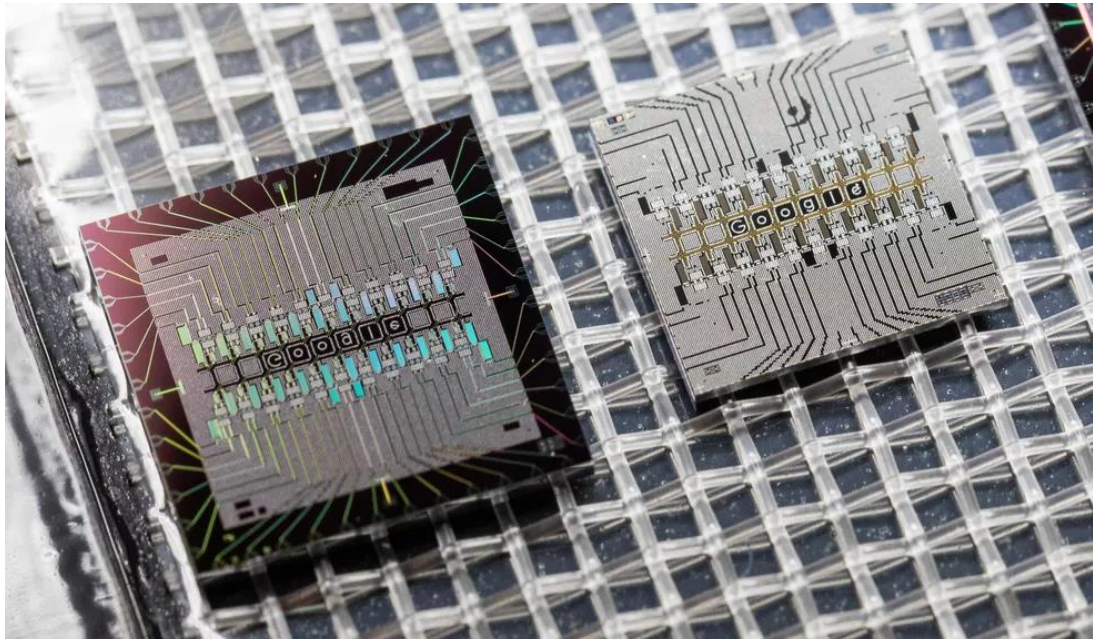
- Arithmetic
- Binary Arithmetic
- Quantum Arithmetic



Introduction

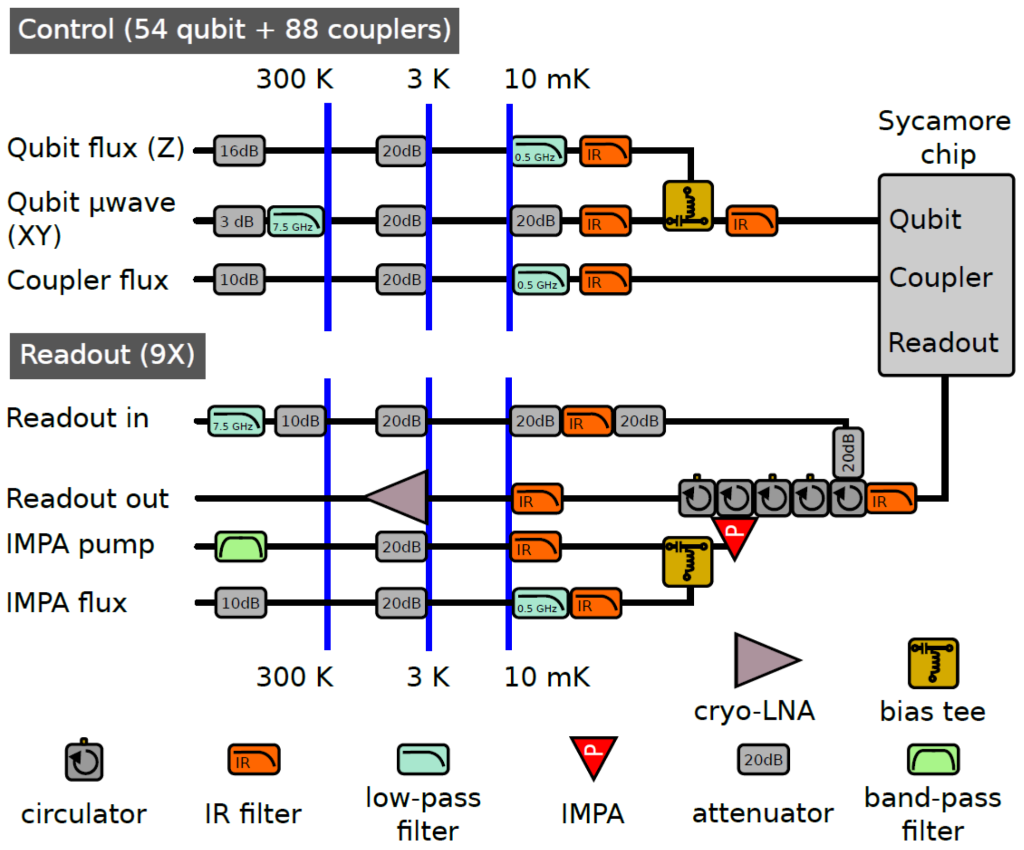
Quantum Computers

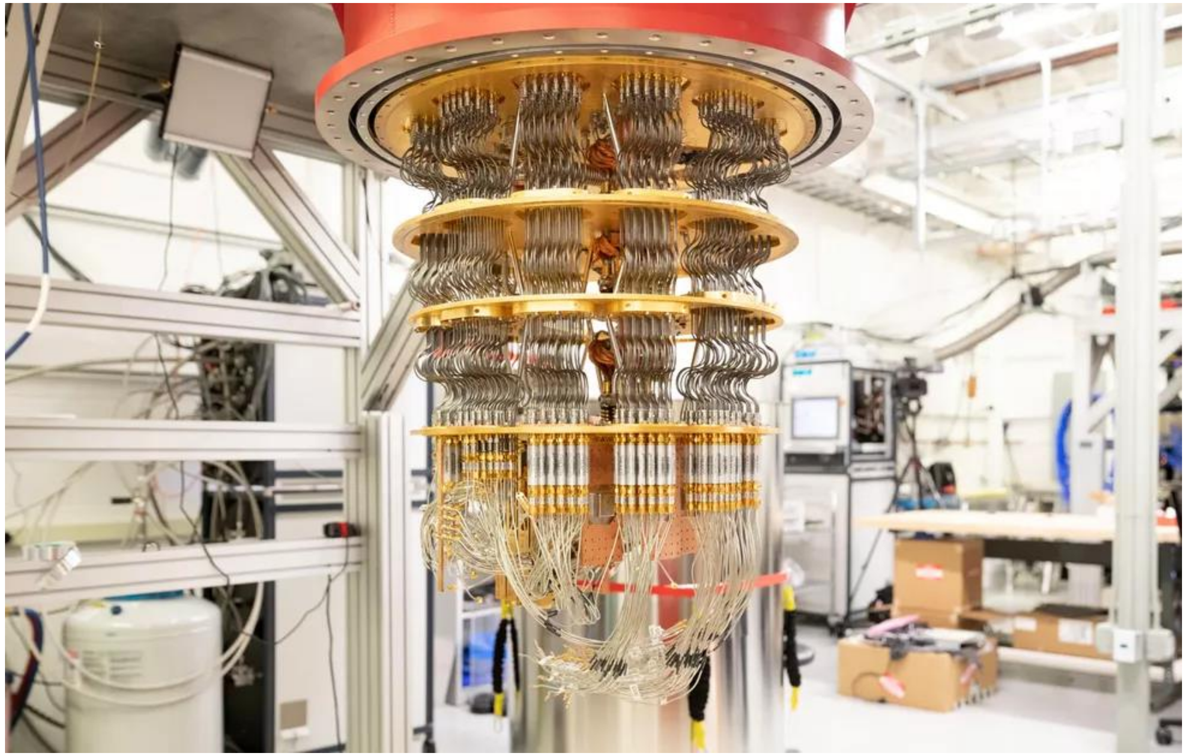




Google Quantum AI

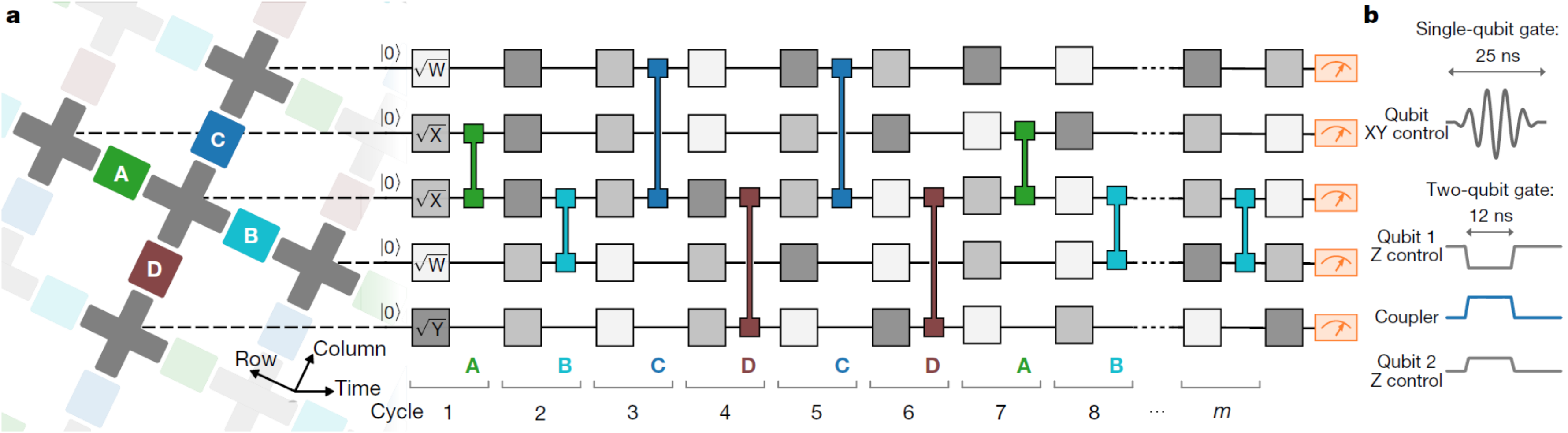
Quantum Computers

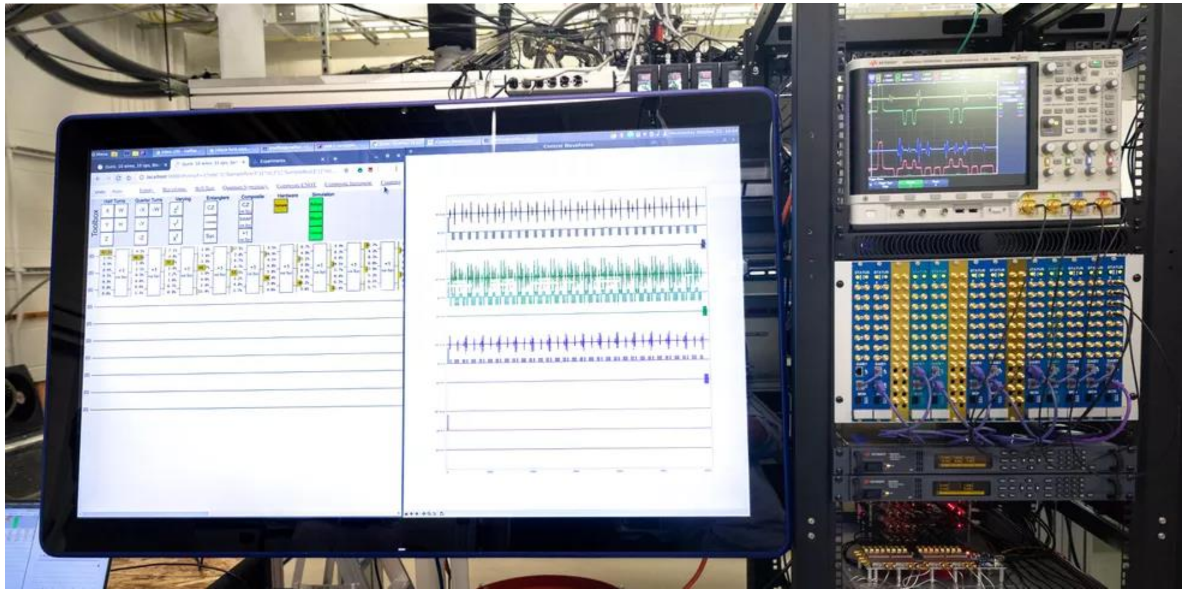




Google Quantum AI

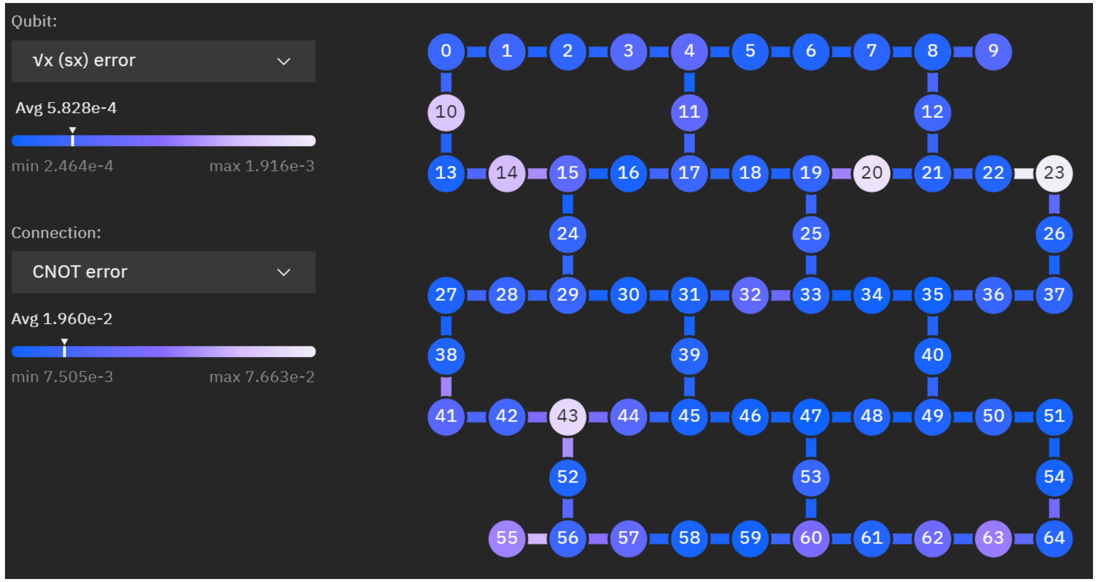
Quantum Computers



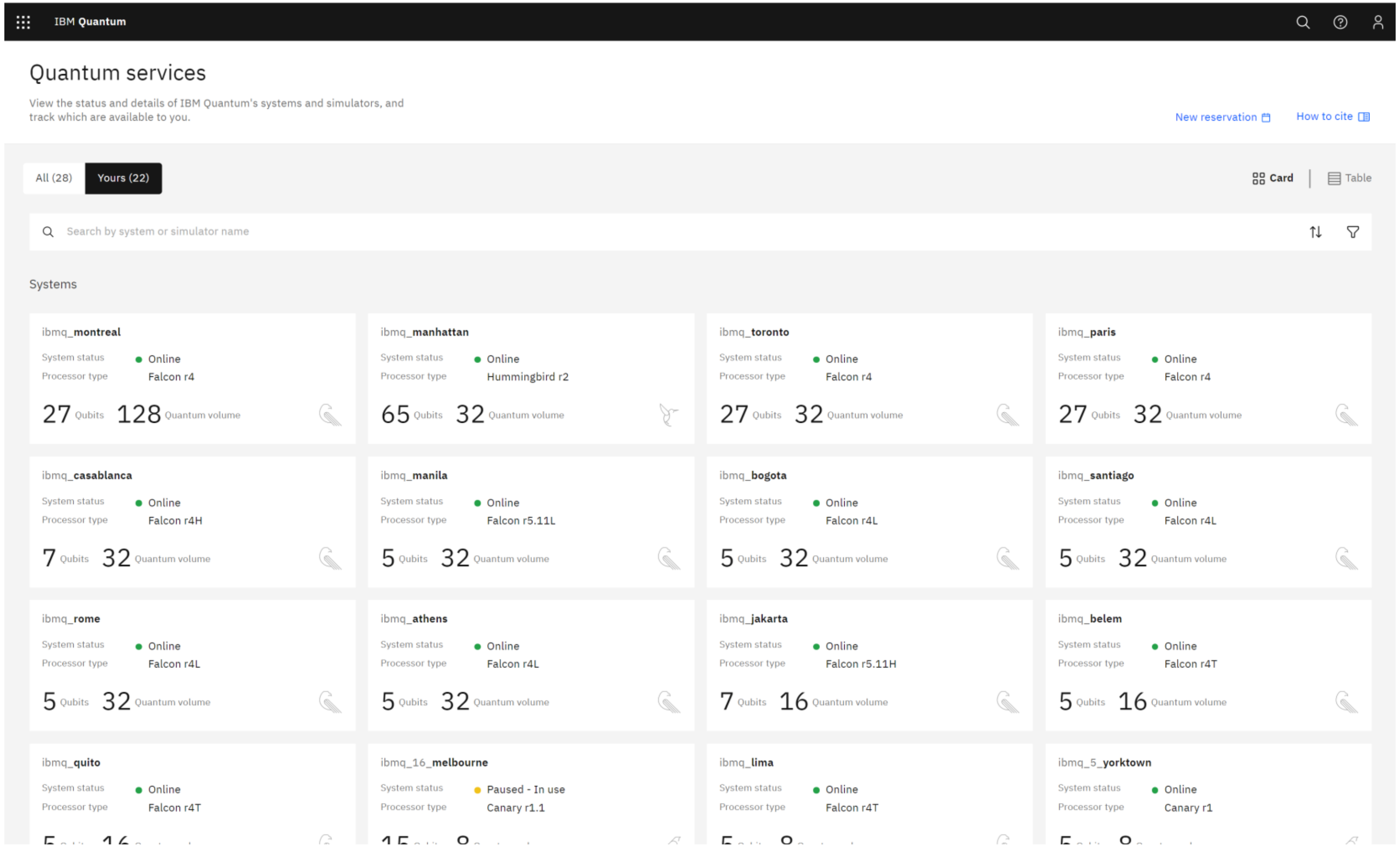


Google Quantum AI

Quantum Computers



Quantum Computers (on the ☁)



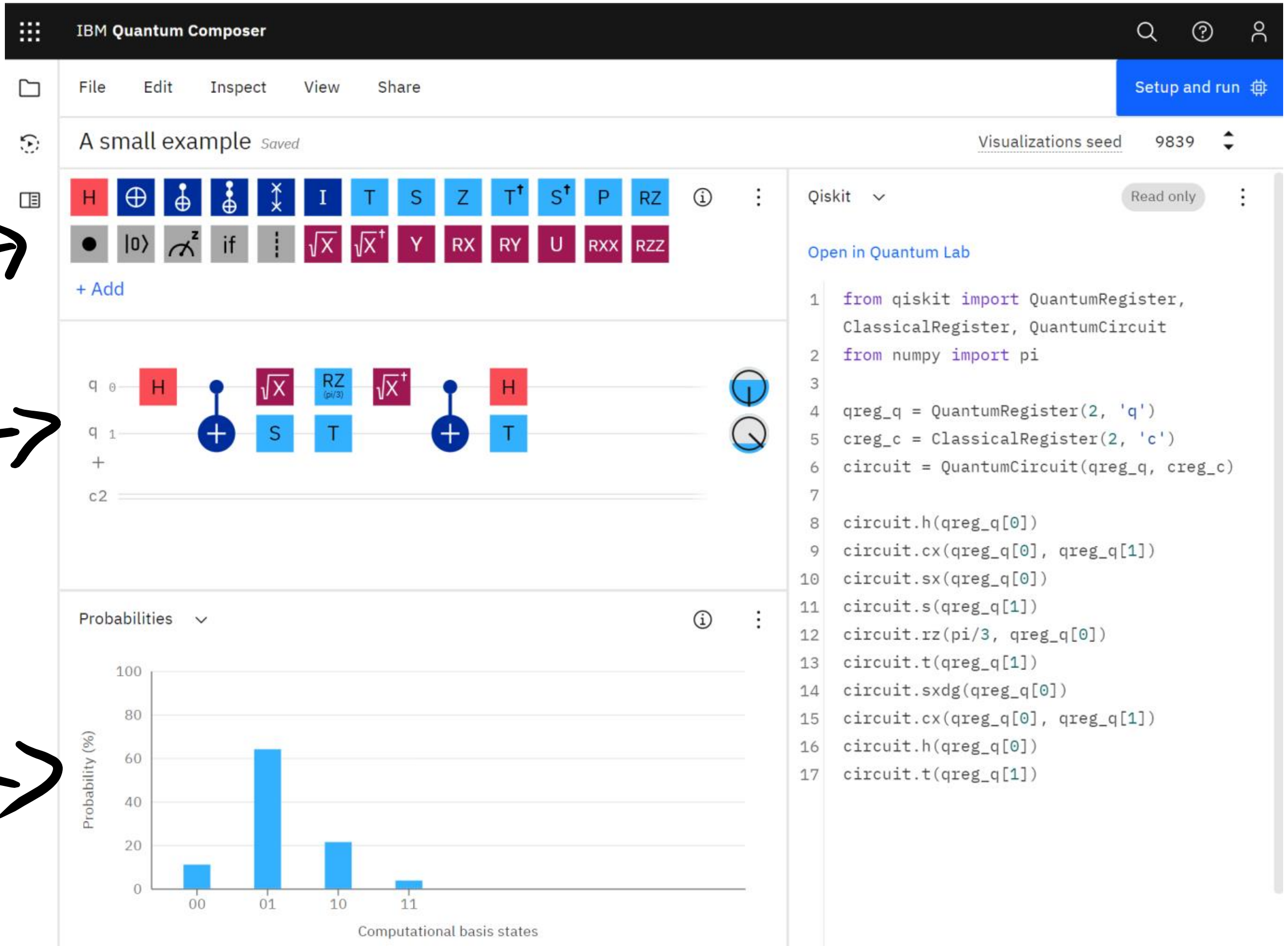
IBM Quantum at https://quantum-computing.ibm.com/

Quantum Computing

quantum gates

quantum circuit

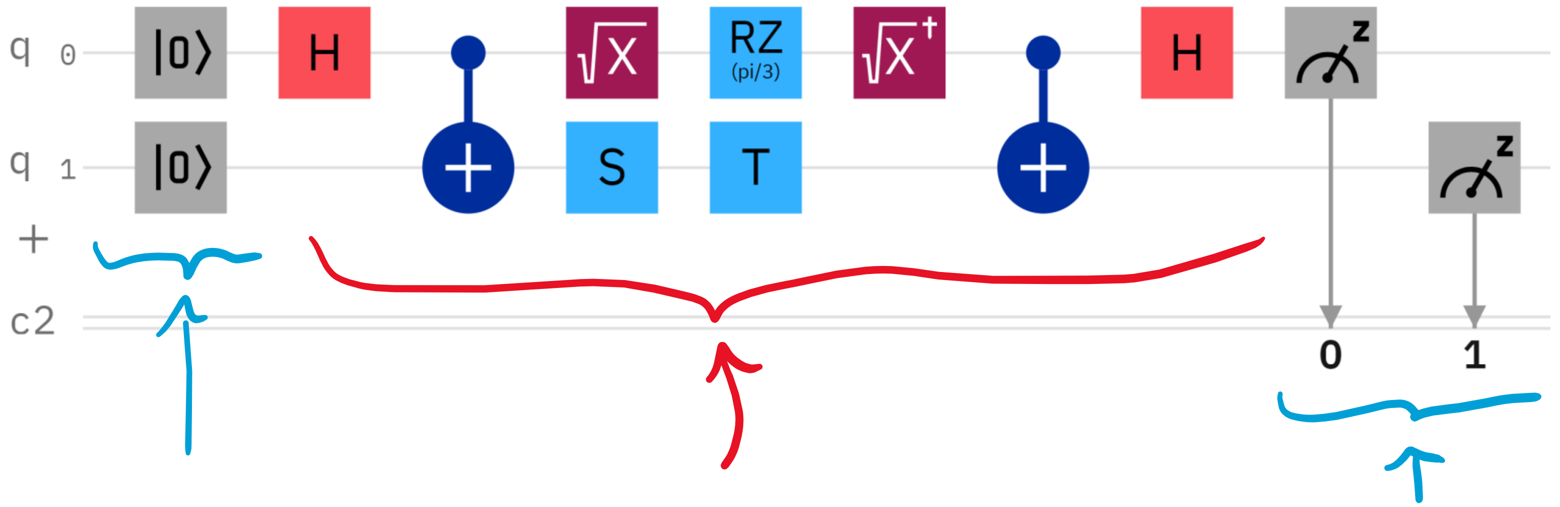
measurement outcomes

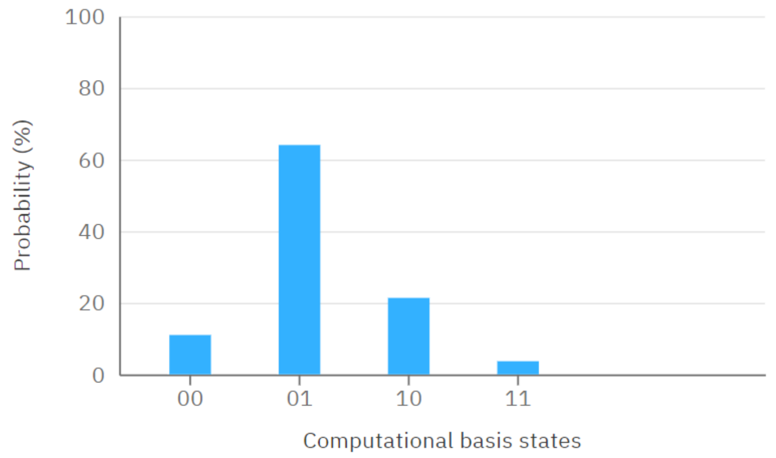


Python code creating the quantum circuit

IBM Quantum at https://quantum-computing.ibm.com/

Quantum Computing





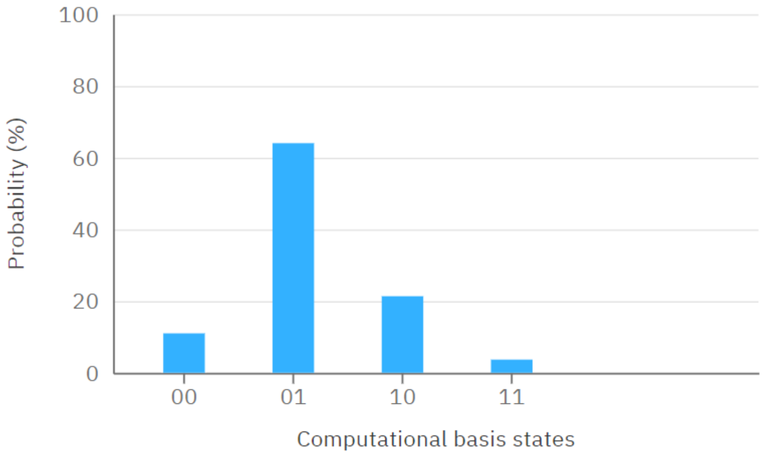
1. Qubits are prepared in a standard $|0\rangle$ state at the start of the computation.

2. Gates are applied to the qubits one after the other, as specified by the circuit.

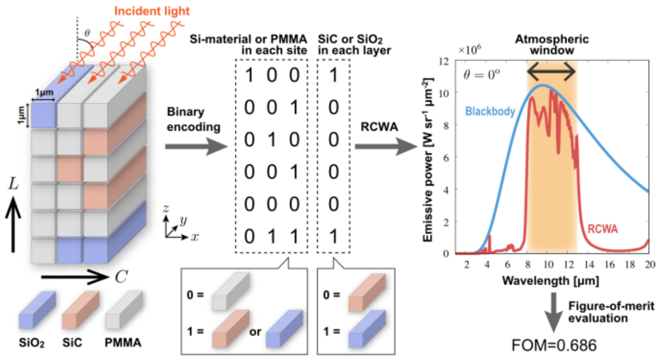
3. Qubits are measured, yielding a bitstring (0 or 1 for each qubit), aka a "measurement outcome"

4. Process is repeated many times and the histogram of observed outcomes is returned.

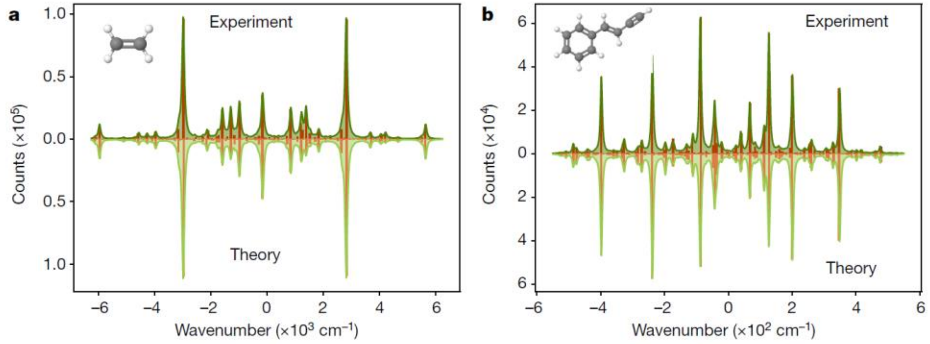
Quantum Computing



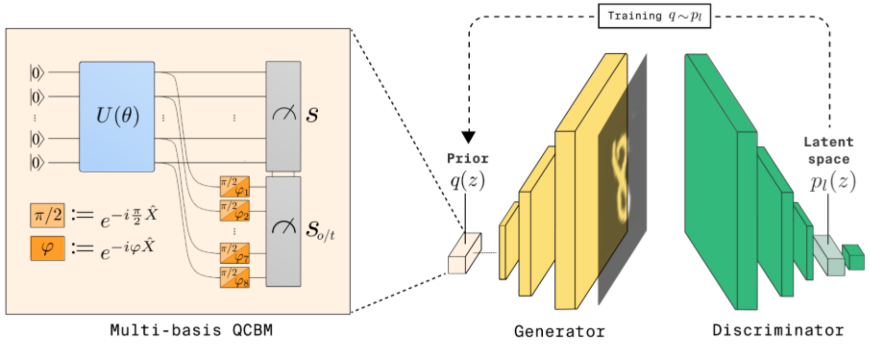
5. The measurement outcome probabilities are post-processed and used to solve interesting real-world problems (usually as part of larger classical algorithms).



Material design



Molecular chemistry

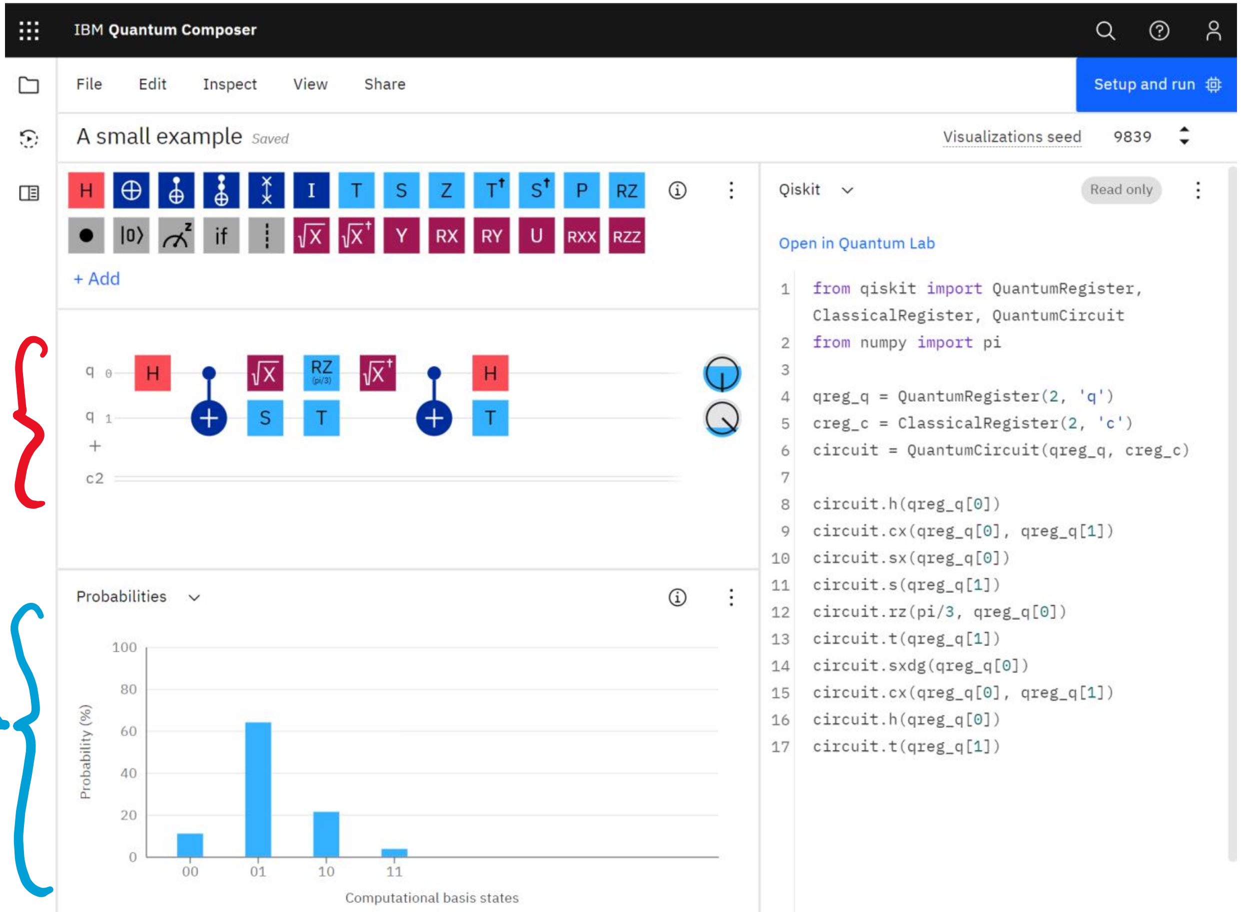


Machine learning

Quantum Computing

Problem

Given a circuit, how do we calculate the probabilities that will be observed from it?

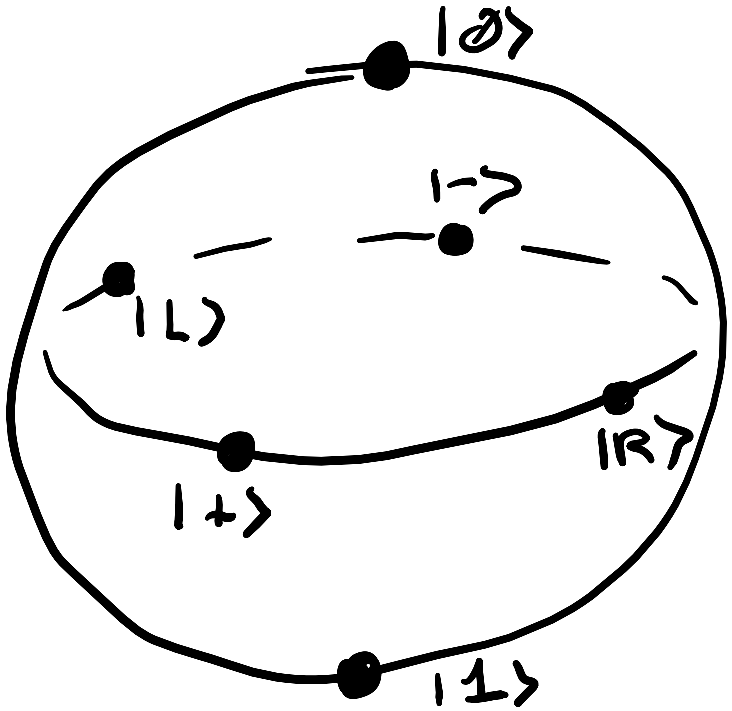


(That is, how do we know which circuits are useful for which problems?)

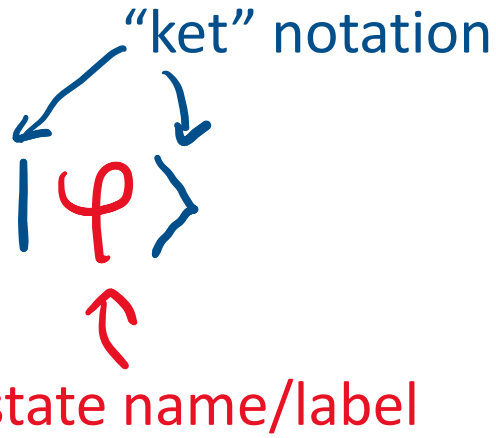
Bits and Qubits

- A bit can take two values: 0 or 1
- A qubit can take infinitely many values: points on a sphere

Known as the "Bloch sphere"

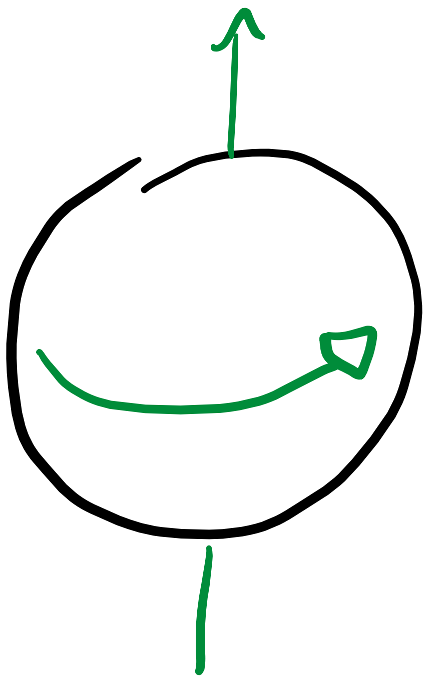


Named quantum states:

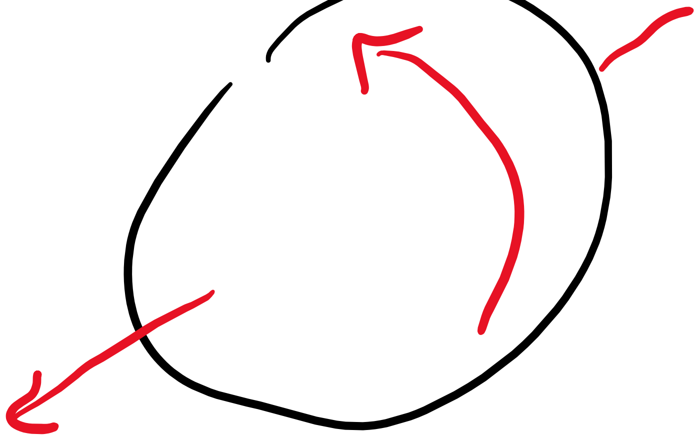


Bits and Qubits

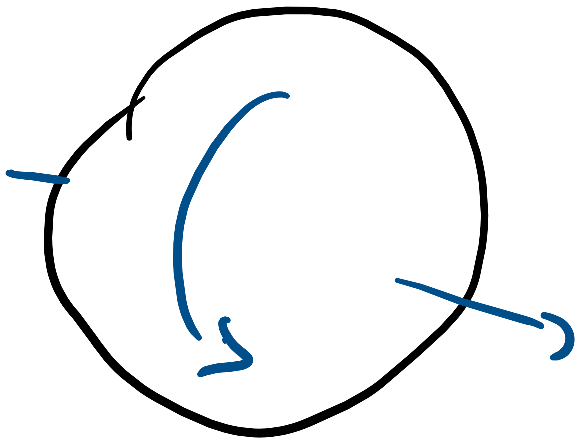
Transformations of a qubit: rotations of the sphere



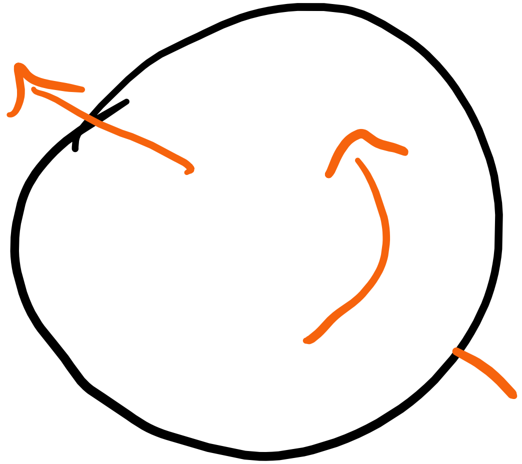
Z-axis rotations



X-axis rotations



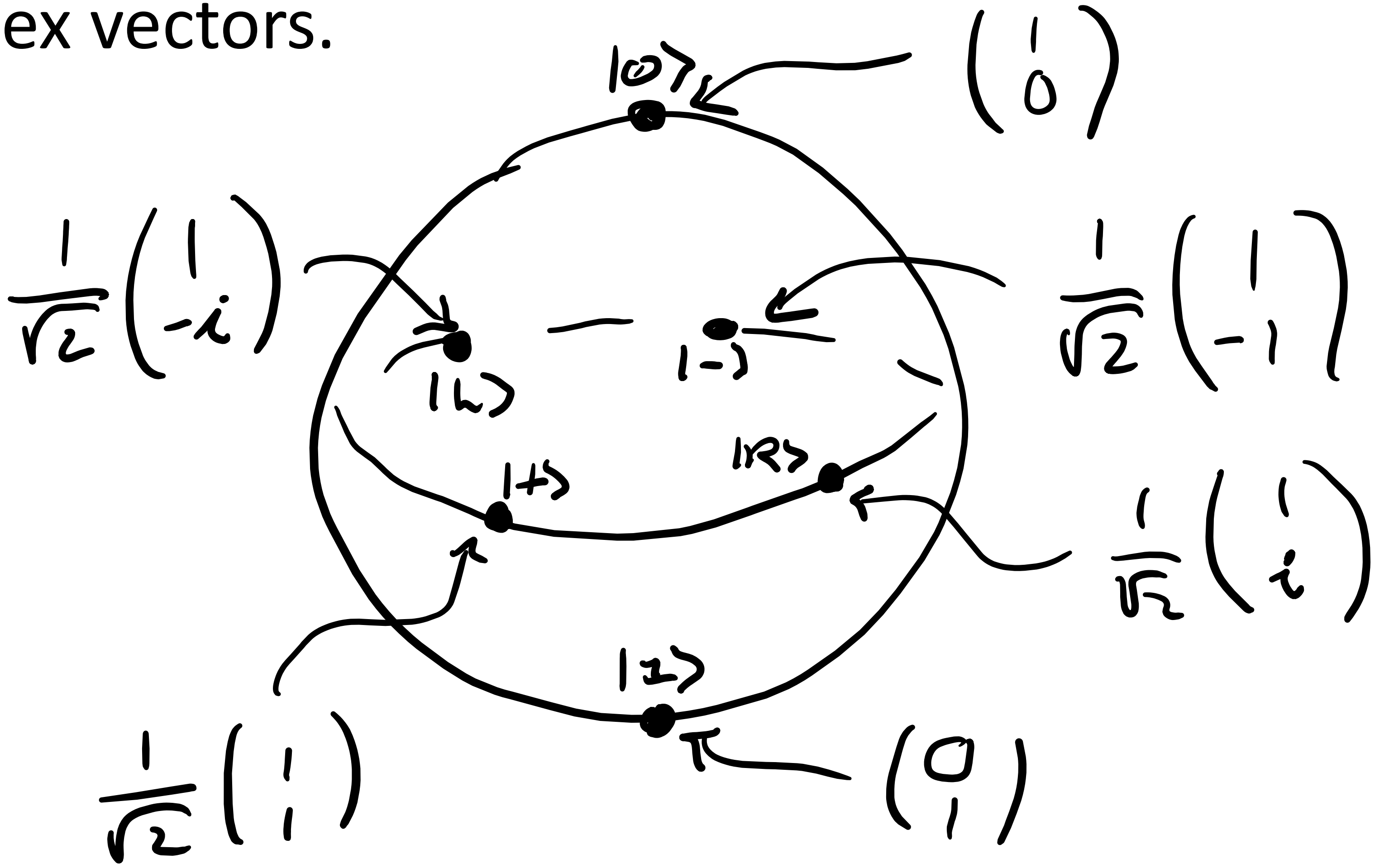
Y-axis rotations



other axis rotations

Bits and Qubits

Qubit values (known as “states”) are traditionally written as complex vectors.



Bits and Qubits

The vectors for an n-qubit state have 2^n components.

$$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -i \end{pmatrix} \qquad \frac{1}{2\sqrt{2}}\begin{pmatrix} 1 \\ -\sqrt{3} \\ \sqrt{3} \\ 1 \end{pmatrix} \qquad \frac{1}{4}\begin{pmatrix} 1 \\ -\sqrt{3} \\ \sqrt{3} \\ 1 \\ -i \\ \sqrt{3}i \\ -\sqrt{3}i \\ -i \end{pmatrix}$$

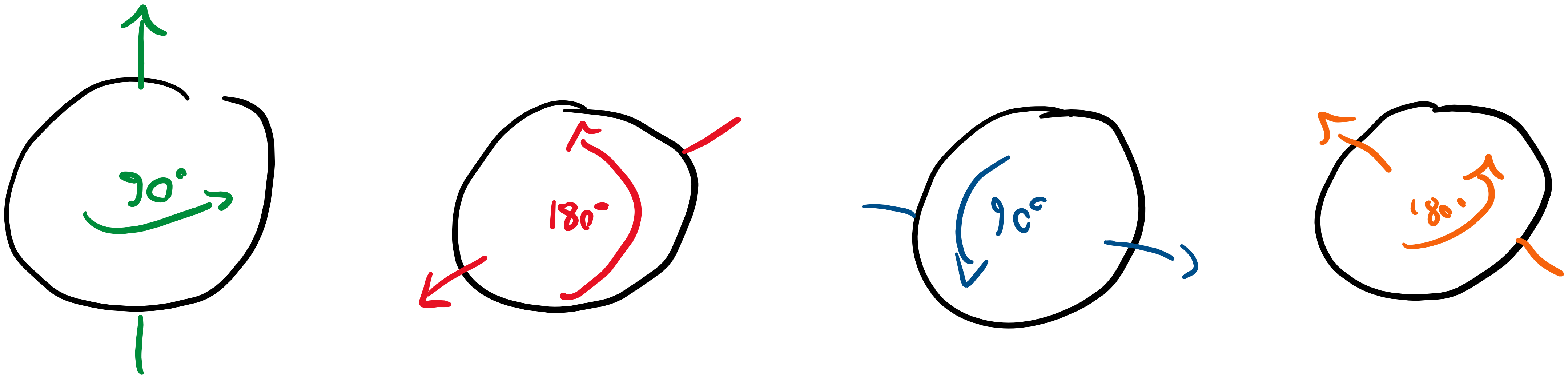
1-qubit state
= 2 components

2-qubit state
= 4 components

3-qubit state
= 8 components

Bits and Qubits

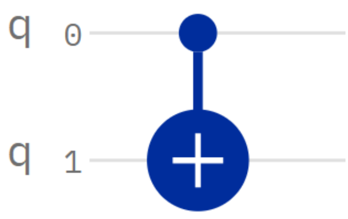
Qubit transformations are then written as complex matrices.

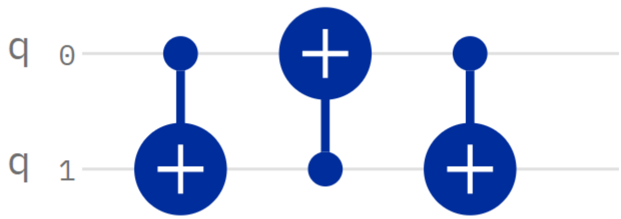


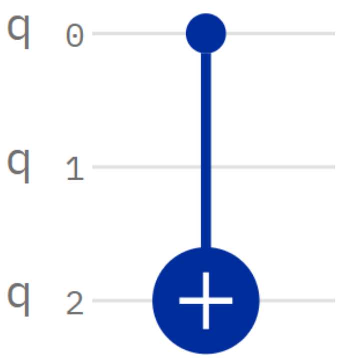
$$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \qquad \frac{}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Bits and Qubits

Qubit transformations are then written as complex matrices.







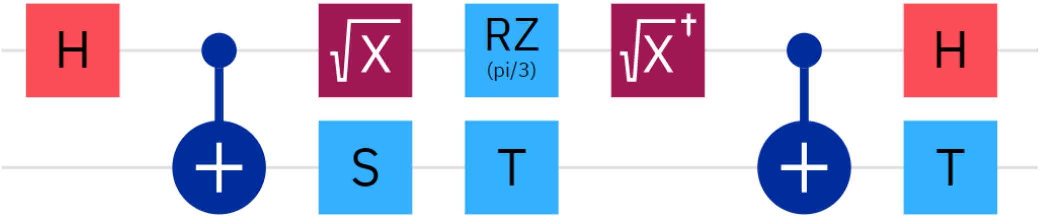
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Bits and Qubits

Qubit transformations are then written as complex matrices.



$$\begin{pmatrix} 0.12+0.30i & 0.17+0.42i & 0.16+0.30i & -0.07-0.17i \\ 0.33-0.30i & -0.17+0.42i & 0.73-0.30i & -0.42+0.17i \\ 0.42+0.17i & 0.30+0.12i & 0.42+0.17i & 0.33+0.30i \\ 0.07-0.17i & -0.30+0.73i & 0.02-0.42i & 0.12-0.30i \end{pmatrix}$$

Bits and Qubits

Vectors and matrices are hard to understand, because they offer no geometrical intuition. Also, they grow exponentially large as more qubits are involved.

Bits and Qubits

Vectors and matrices are hard to understand, because they offer no geometrical intuition. Also, they grow exponentially large as more qubits are involved.

For 1 qubit, the sphere provides a nice geometric model.

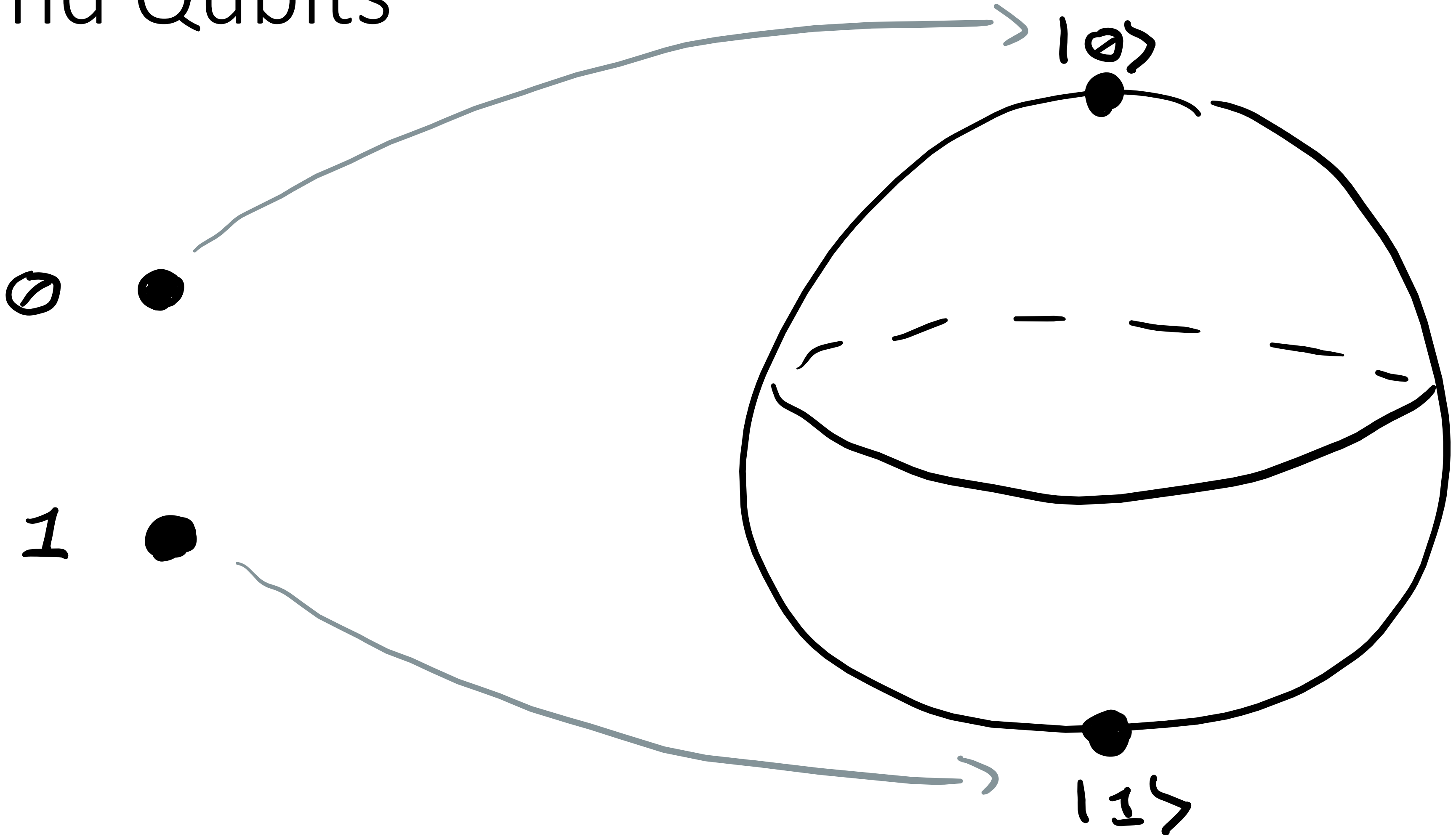
Bits and Qubits

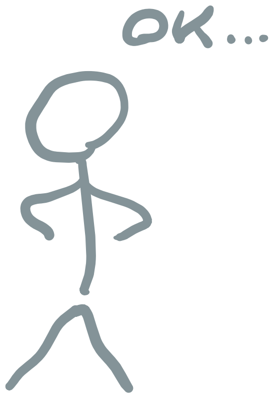
Vectors and matrices are hard to understand, because they offer no geometrical intuition. Also, they grow exponentially large as more qubits are involved.

For 1 qubit, the sphere provides a nice geometric model.

For 2+ qubits, unfortunately, things get really complicated, really fast: geometrical intuition is better left to trained mathematicians…

Bits and Qubits

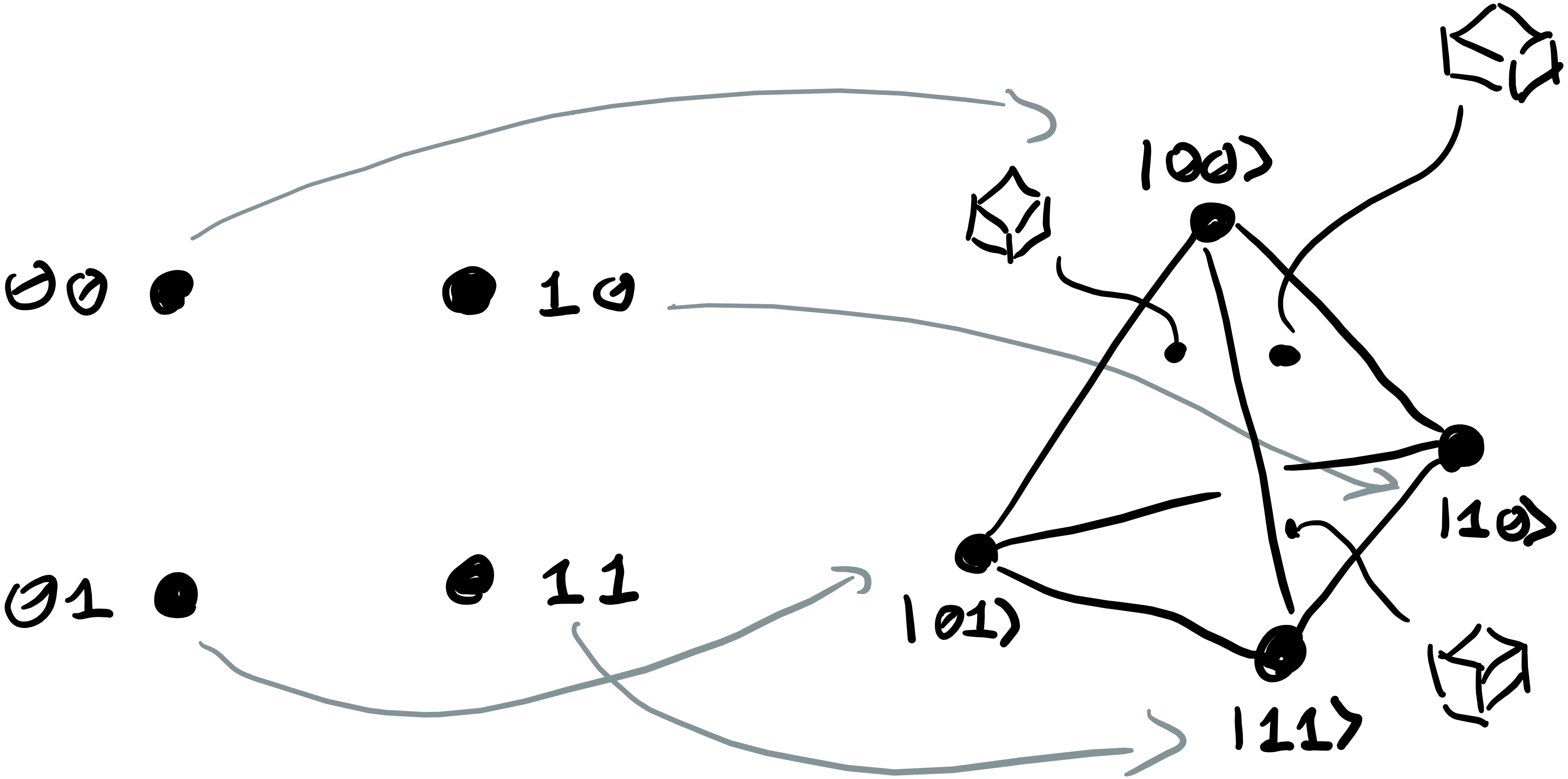




1 bit = 2 points

1 qubit = a hollow sphere

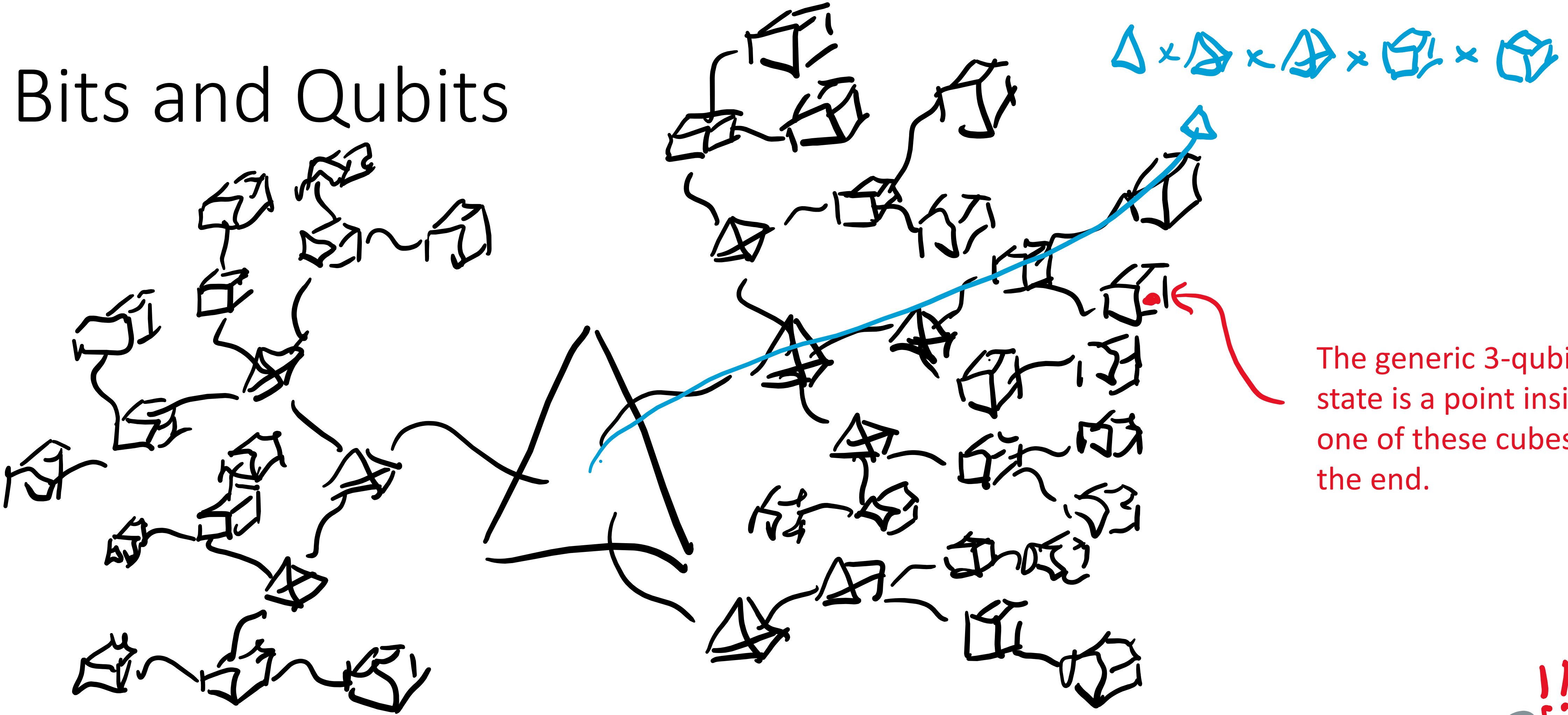
Bits and Qubits



2 bits = 4 points

2 qubits = a filled pyramid where each point is a filled cube.

Bits and Qubits



The generic 3-qubit state is a point inside one of these cubes at the end.

3 qubits = a filled triangle where each point is a filled pyramid, where each point is another filled pyramid, where each point is a filled cube, where each point is another filled cube.

A change of perspective

The traditional representation of qubit states and their transformations as complex vectors and matrices is not great:

- gives no intuition => not useful to understand/design circuits
- grows exponentially large => not useful for calculations (by hand)

A change of perspective

The traditional representation of qubit states and their transformations as complex vectors and matrices is not great:

- gives no intuition => not useful to understand/design circuits
- grows exponentially large => not useful for calculations (by hand)

Furthermore, the geometric representation is only useful for single-qubit states and transformations (the Bloch sphere).

A change of perspective

The traditional representation of qubit states and their transformations as complex vectors and matrices is not great:

- gives no intuition => not useful to understand/design circuits
- grows exponentially large => not useful for calculations (by hand)

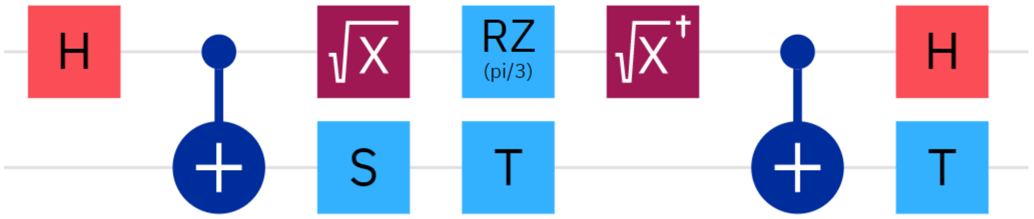
Furthermore, the geometric representation is only useful for single-qubit states and transformations (the Bloch sphere).

What do we do? We need a change of perspective.

A change of perspective

Quantum circuits make it easy to understand what's happening to the qubits in a quantum computer.

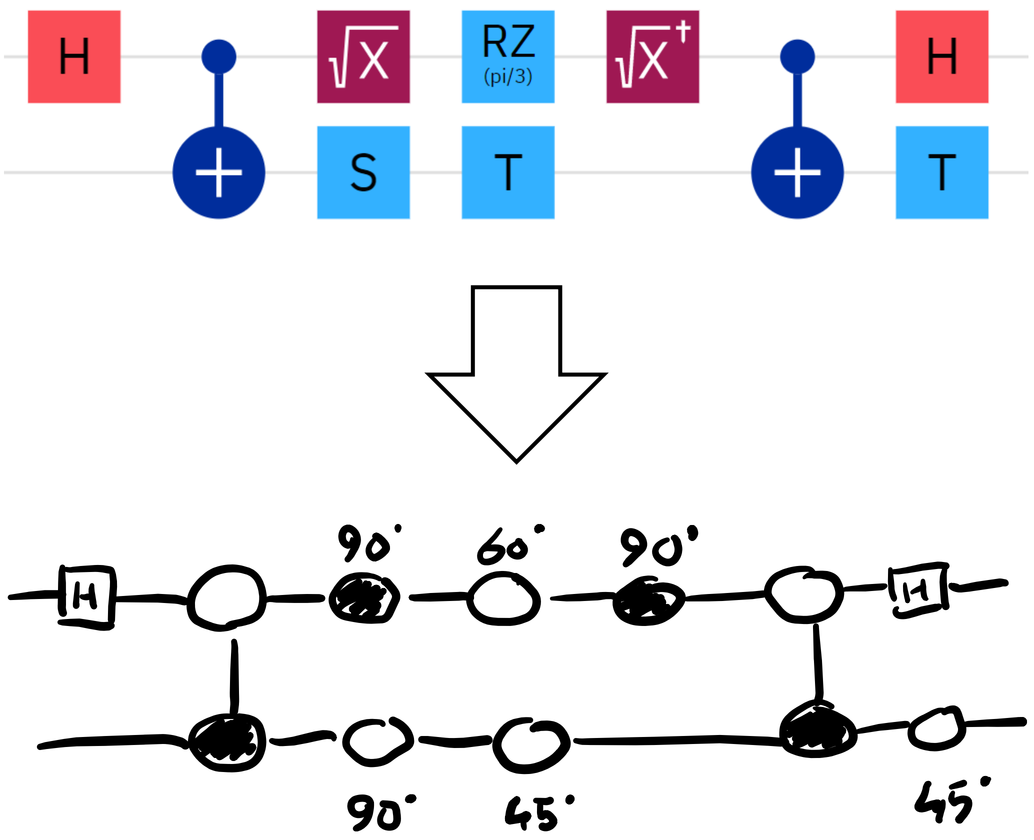
Unfortunately, we can't use them directly to perform calculations.



A change of perspective

<u>Solution</u>

Transform quantum circuits into another kind of diagrams, with simple graphical rules that can be used to perform calculations.



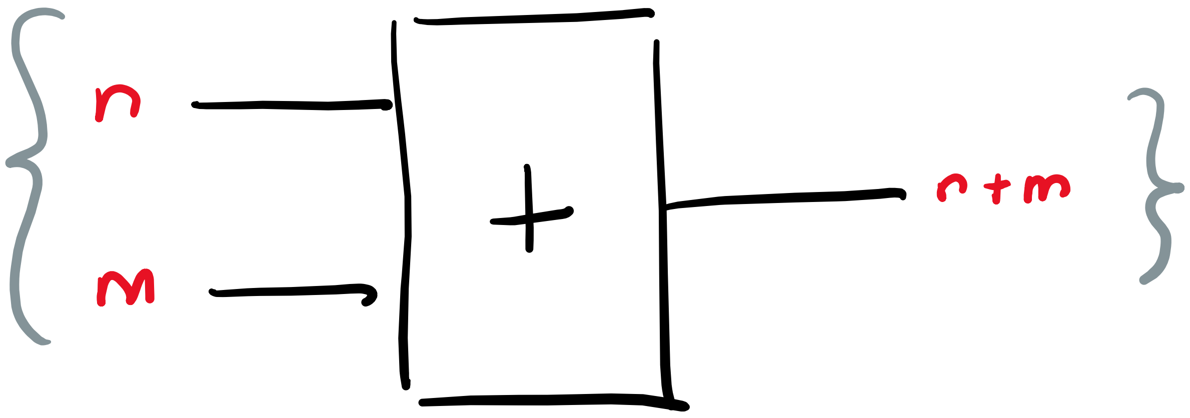
Let’s take a short break (10m)

Arithmetic with Pictures

Gates for arithmetic

Addition as a gate:

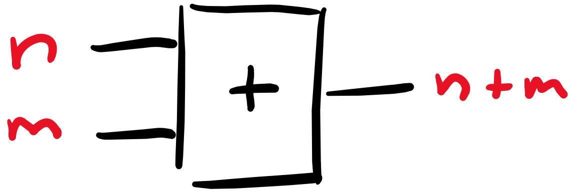
2 numbers go
into the gate
(its “inputs”)



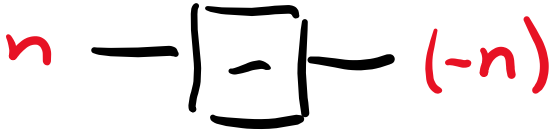
1 number
comes out of
the gate
(its “output”)

Gates for arithmetic

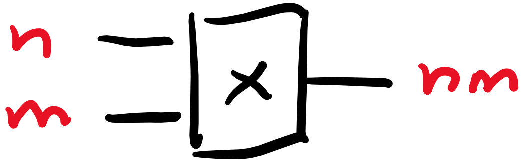
Addition:



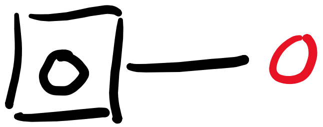
Negation:



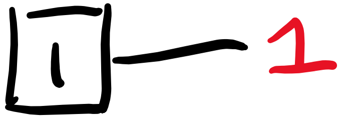
Multiplication:



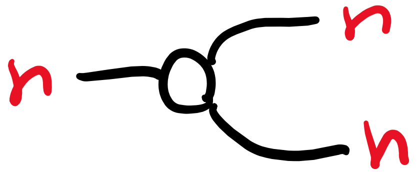
Prepare 0:



Prepare 1:



Copy:

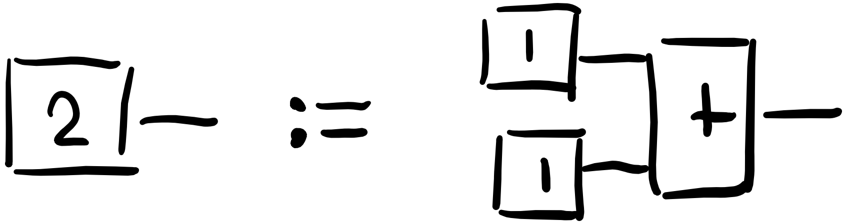


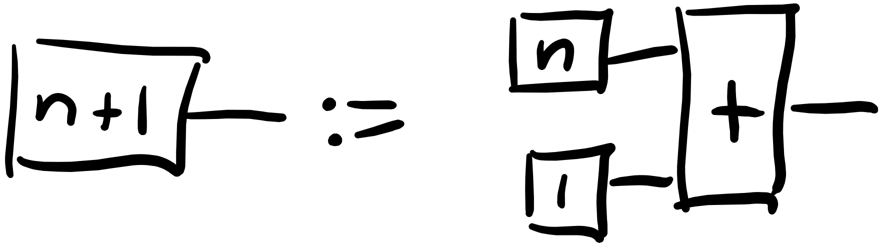
Discard:

Gates for arithmetic

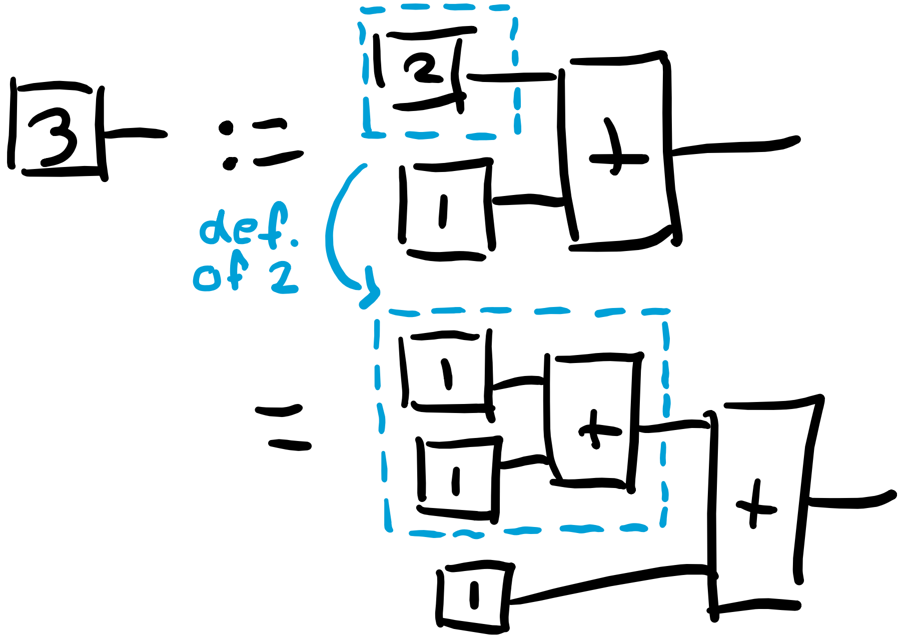
Preparing 0 and 1 is enough: all other numbers can be prepared by applying addition gates to enough copies of 1.

$2 := 1+1$



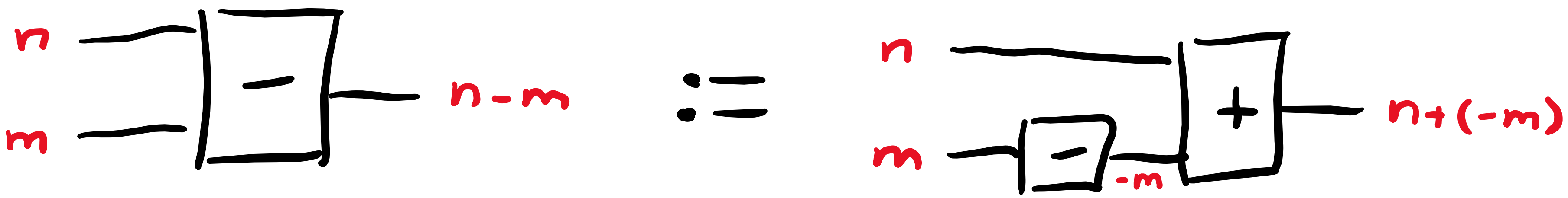


$3 := 2+1 = (1+1)+1$



Gates for arithmetic

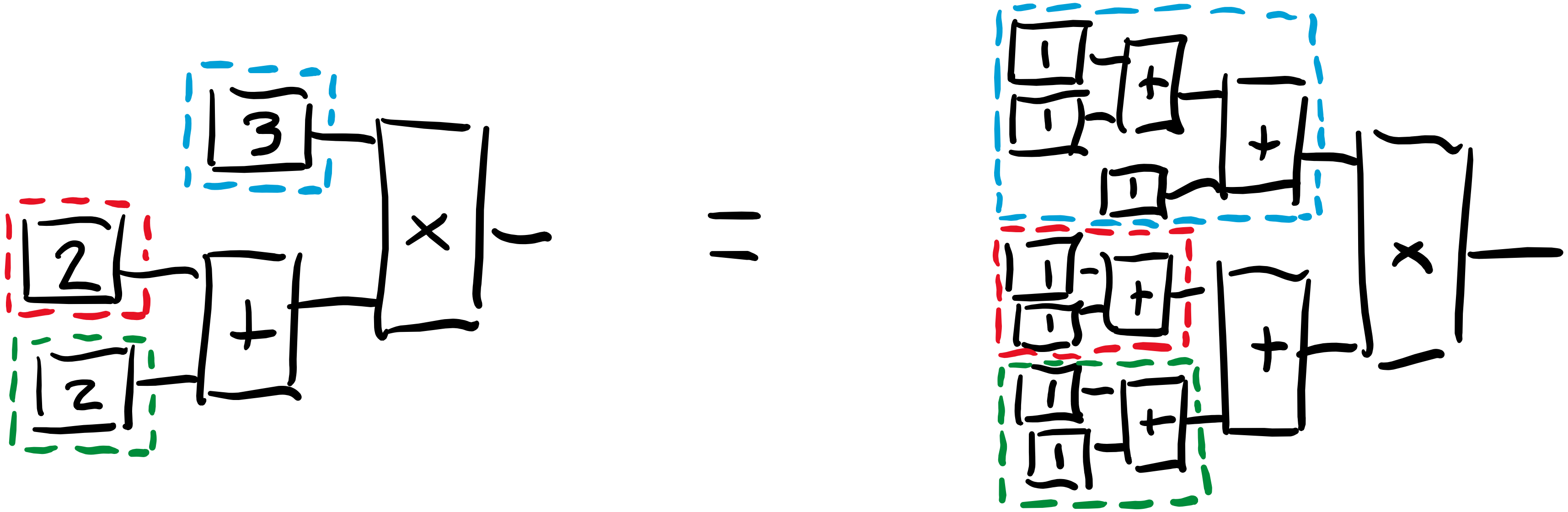
We didn’t write a subtraction gate, because it can be defined in terms of addition and negation:



We have two gates labelled by “-”, but they cannot be confused: one takes two inputs (subtraction), the other takes one (negation).

Gates for arithmetic

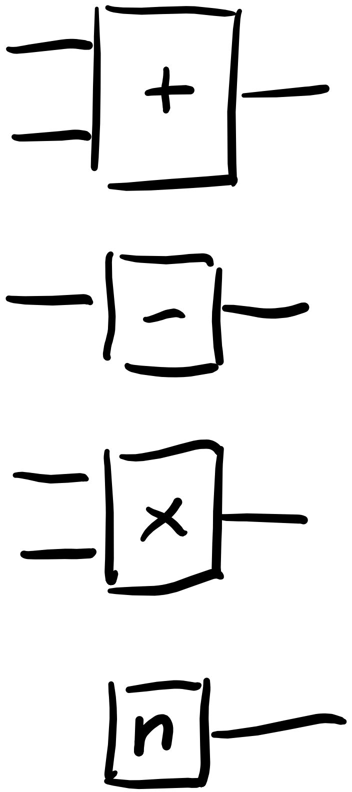
The gates we defined so far are enough to translate all arithmetic expressions as circuits. For example:



3 x (2 + 2)

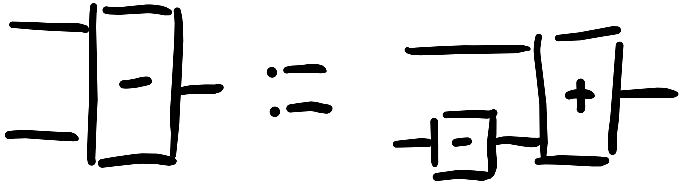
((1+1)+1) x ((1+1) + (1+1))

Exercise (5m)



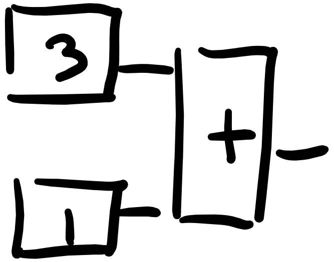
Using the gates on the left and the definition below for subtraction, write the following expressions as circuits:

- 3+1
- (-2)+1
- (1+2)+1
- (2+1)x(2-1)

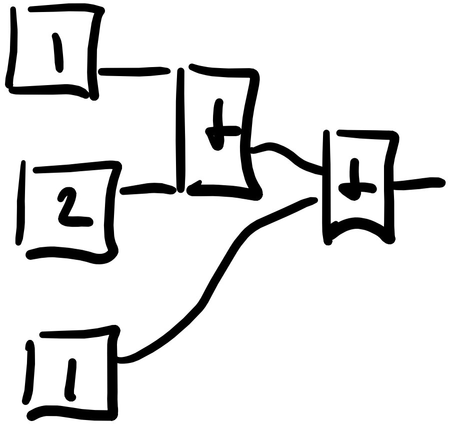


Exercise (Solutions)

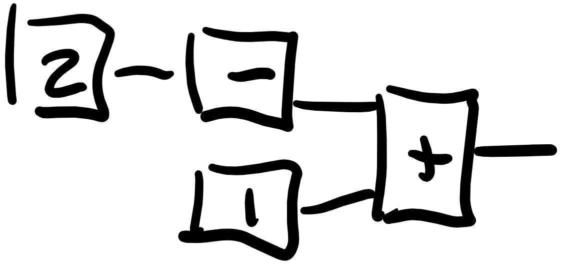
3+1



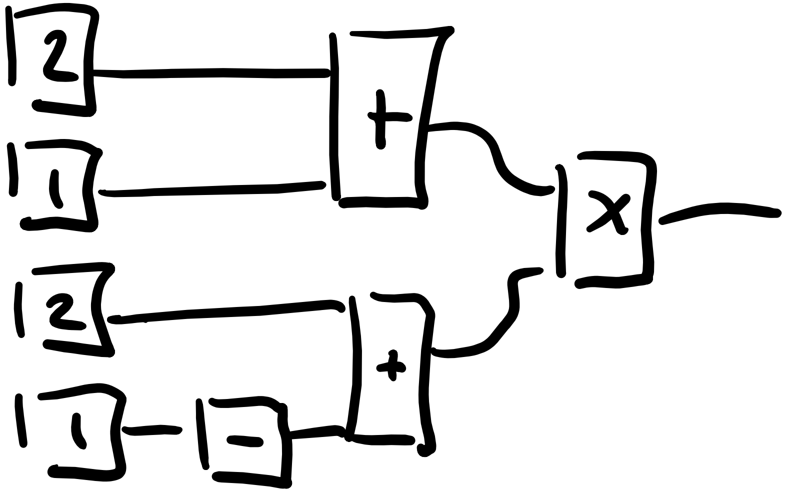
(1+2)+1



(-2)+1

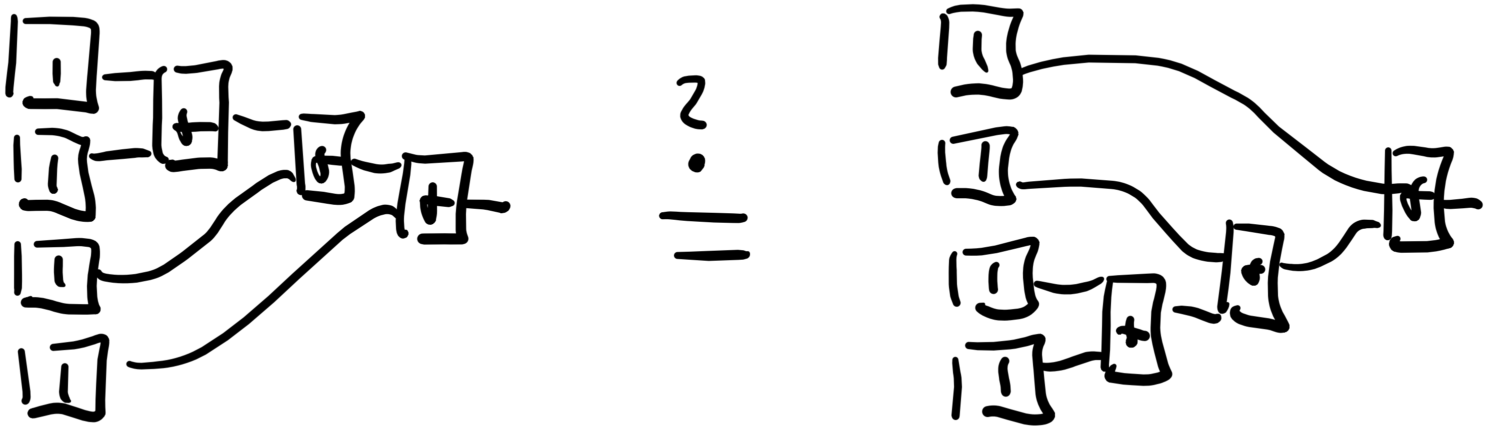


(2+1)x(2-1)



Rules of arithmetic

With gates we can write expressions, but we cannot perform calculations. For example, the following two circuits give the same number at the end, but we have no way to equate them:



3+1 1+(1+2)

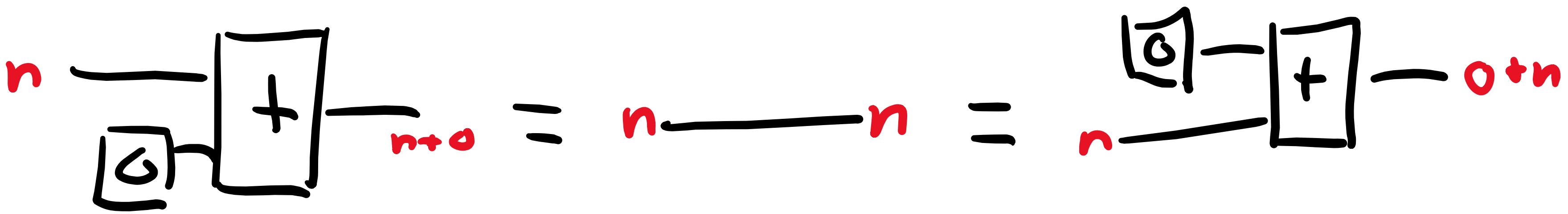
Rules of arithmetic

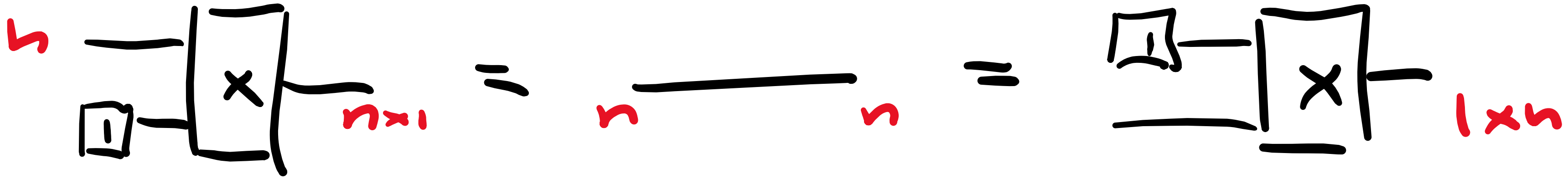
In order to perform calculations, we need “rules” to tell us how we can turn a circuit into an “equivalent” one, that is, without changing its result.

Luckily, you already know (some of) these rules: they are the Laws of Arithmetic!

Neutral elements

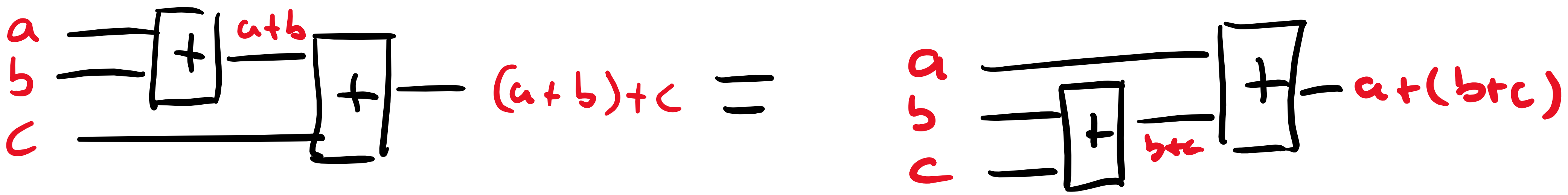
Neutral elements give a way to remove gates from a circuit:



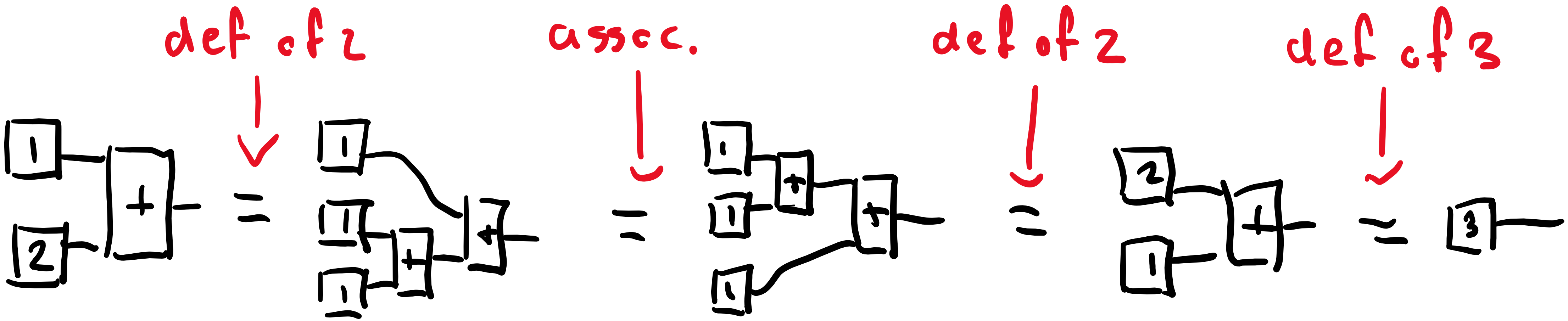


Associativity

Associativity allows us to reorder addition gates in sequence:

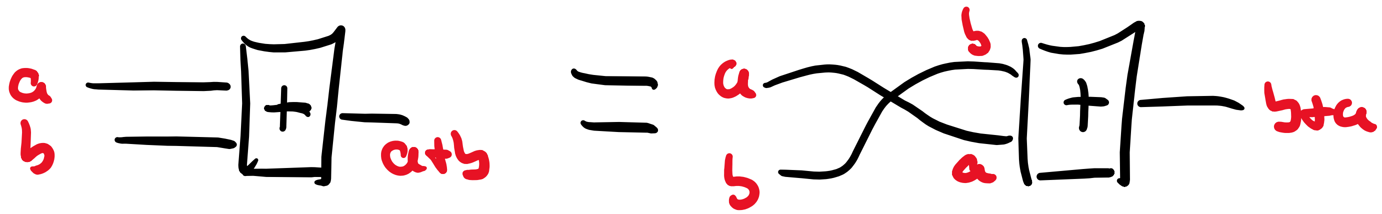


Computation by addition



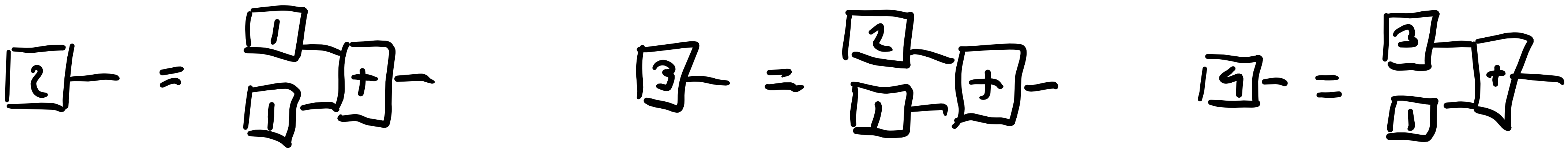
Commutativity

Commutativity allows us to change the order of inputs:



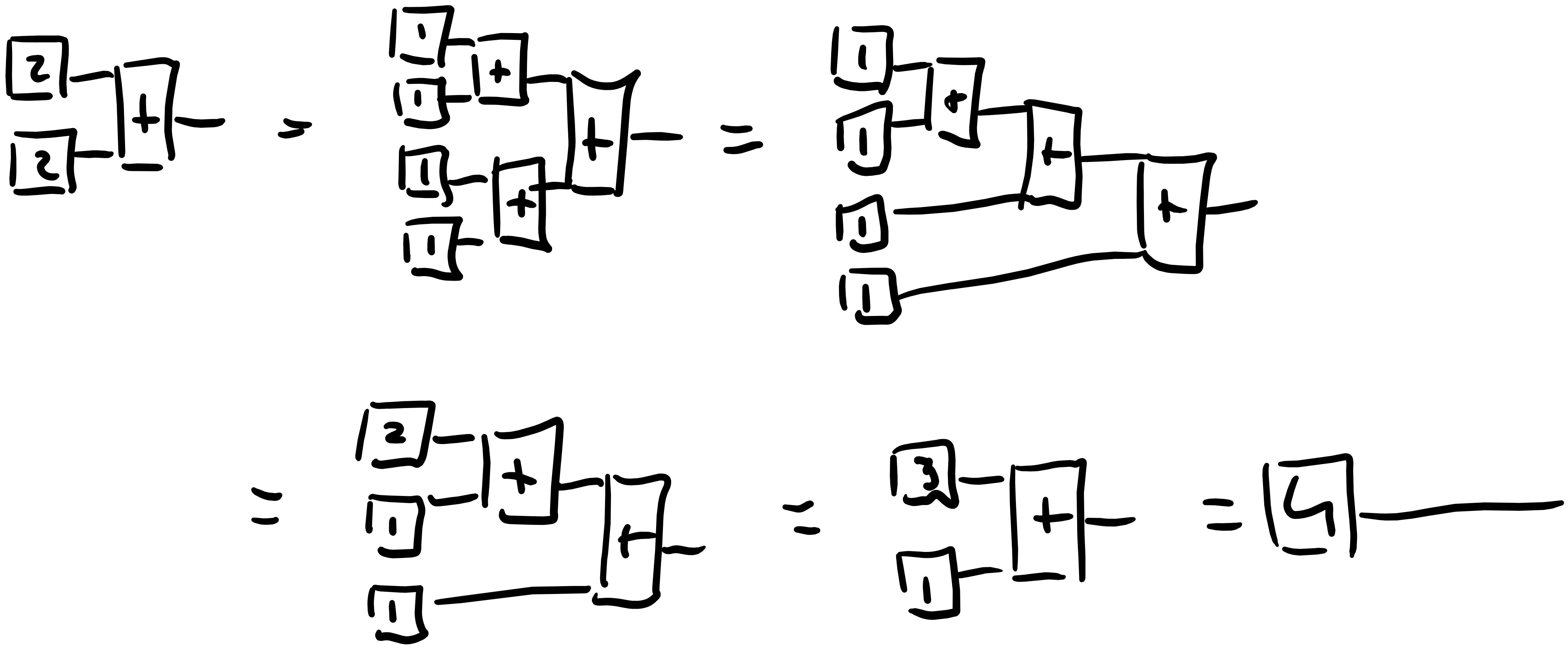
Exercise (5m)

Prove that 2+2=4, using associativity and the definitions of the numbers 2, 3 and 4:



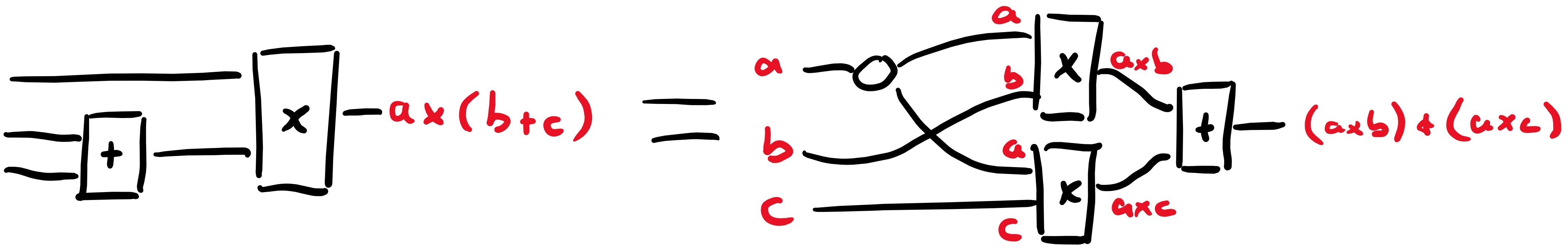
Exercise (Solutions)

Prove that 2+2=4, using associativity and the def’s of 2 and 4.

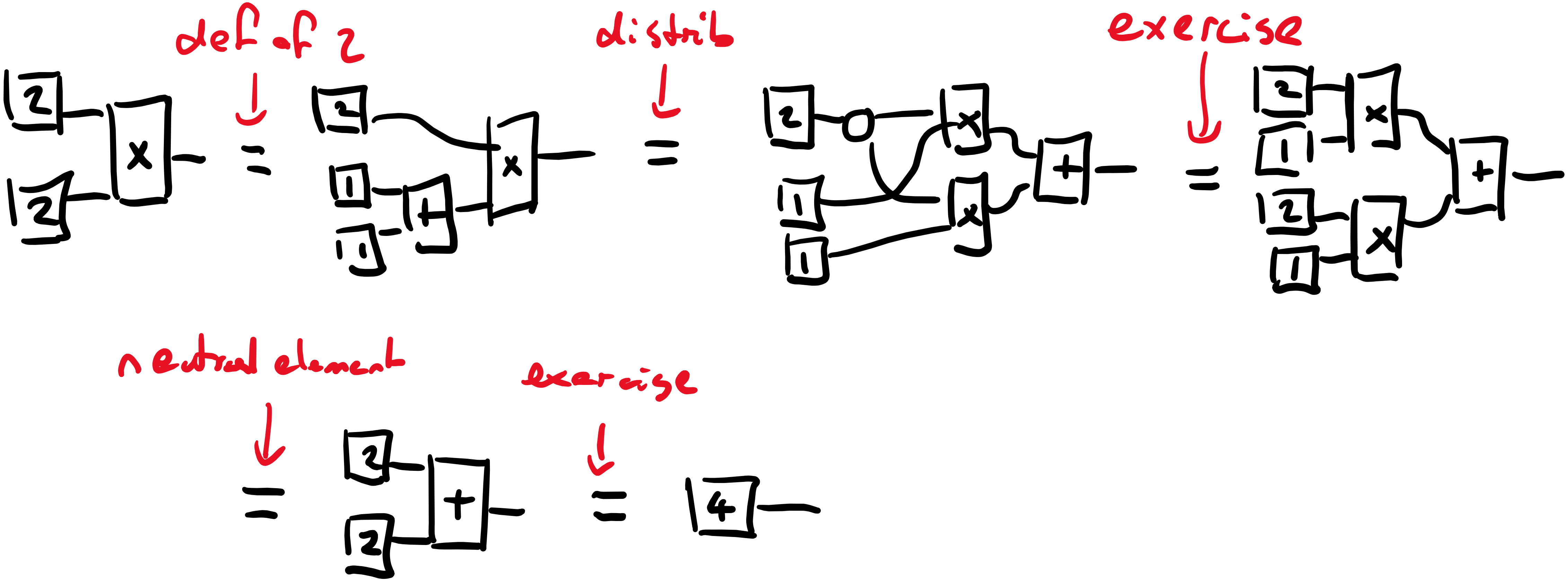


Distributivity

Distributivity allows us to reorder addition and multiplication gates in sequence:

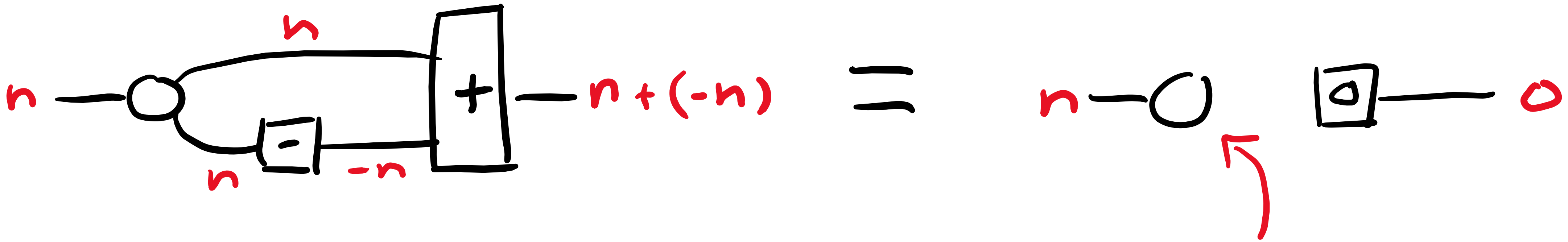


Computation by multiplication



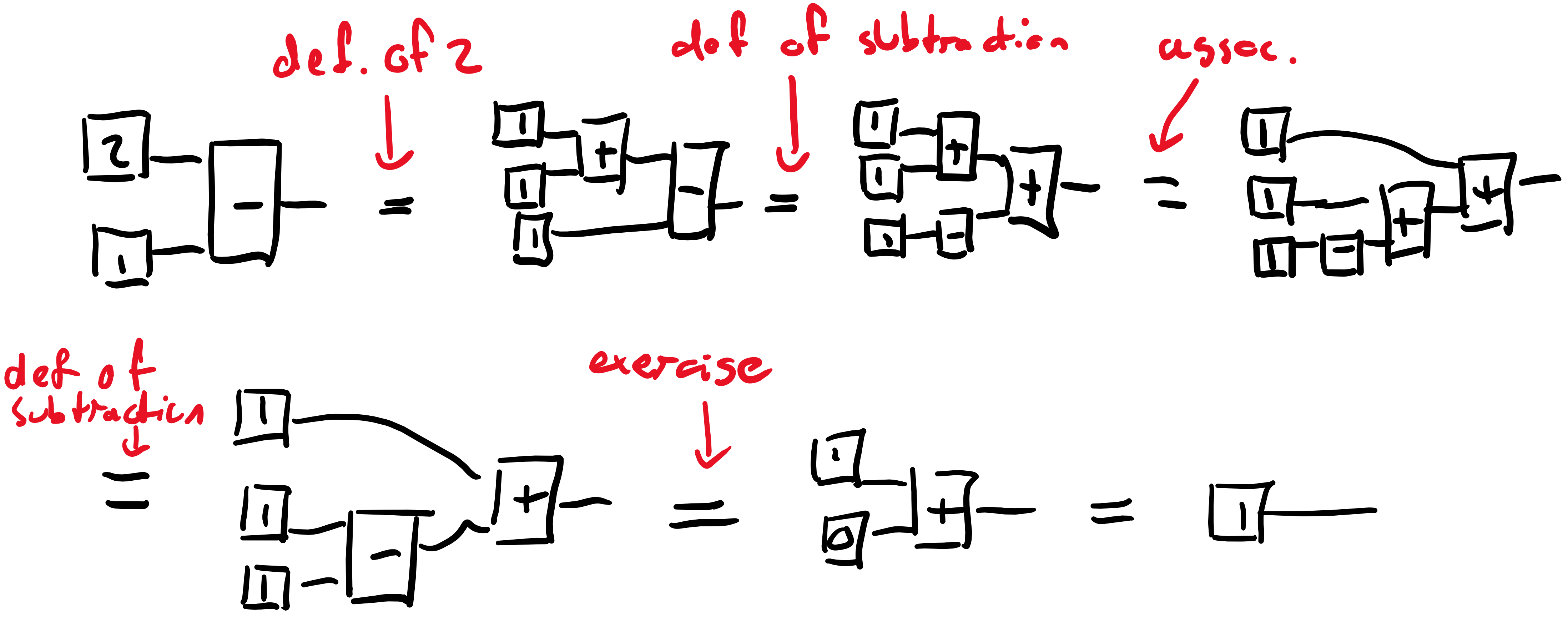
Cancellativity

Cancellativity is the first rule where deleting becomes necessary, because n does not appear in the right hand side of the equation n+(-n) = 0.



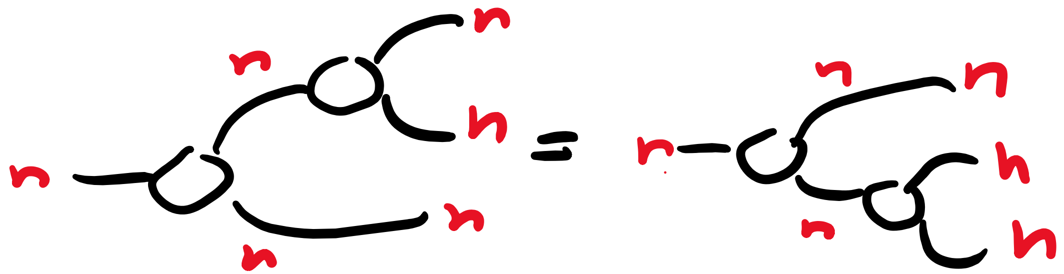
n is removed from the equation here

Computation by subtraction

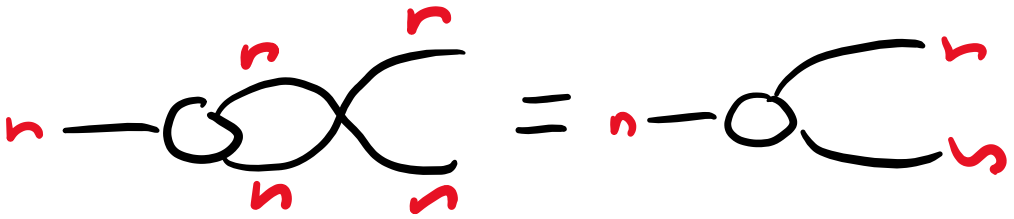


Rules of copying

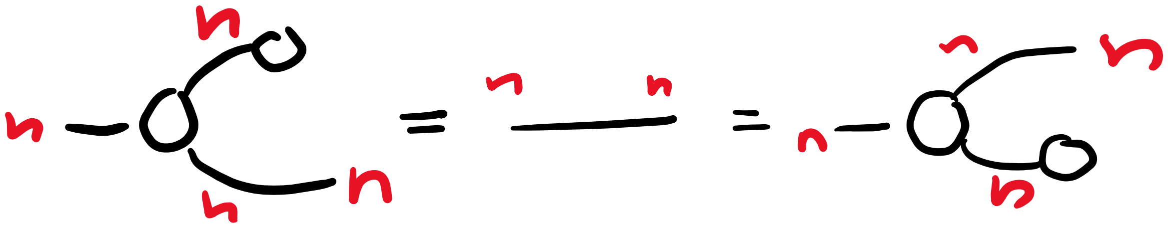
Associativity:



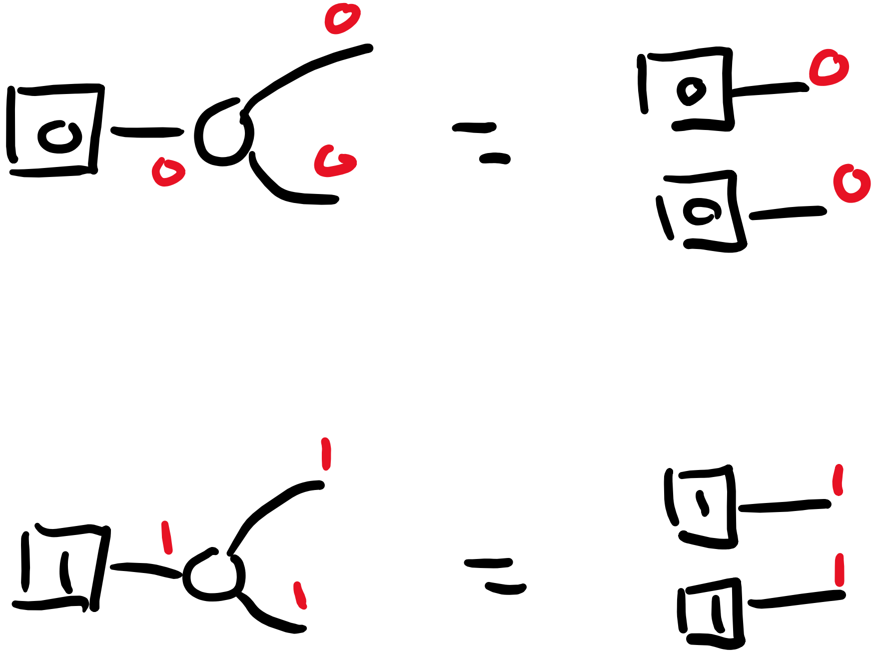
Commutativity:

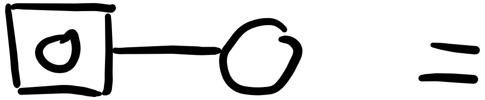


Neutral element (deletion):

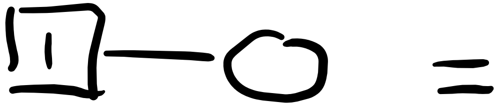


Copy/delete rules for states

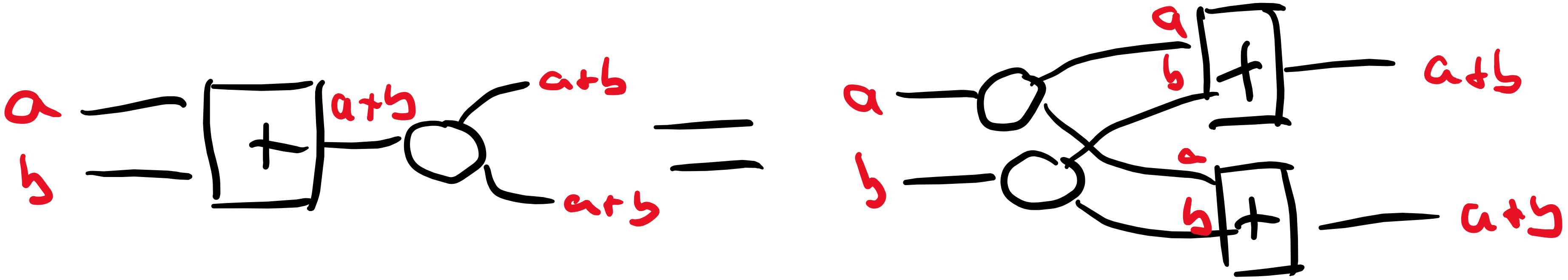


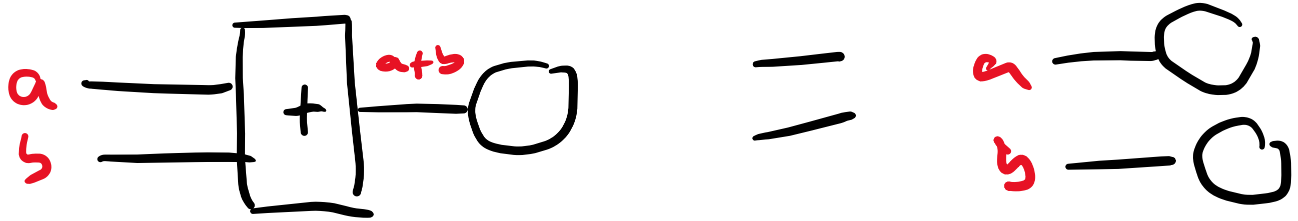


Right hand side is empty because
the state is removed entirely



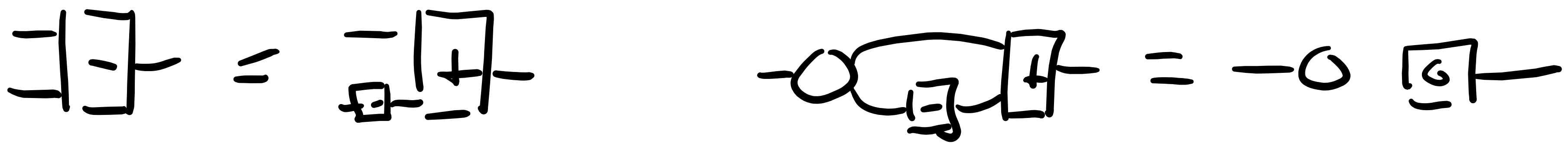
Copy/delete rules for gates





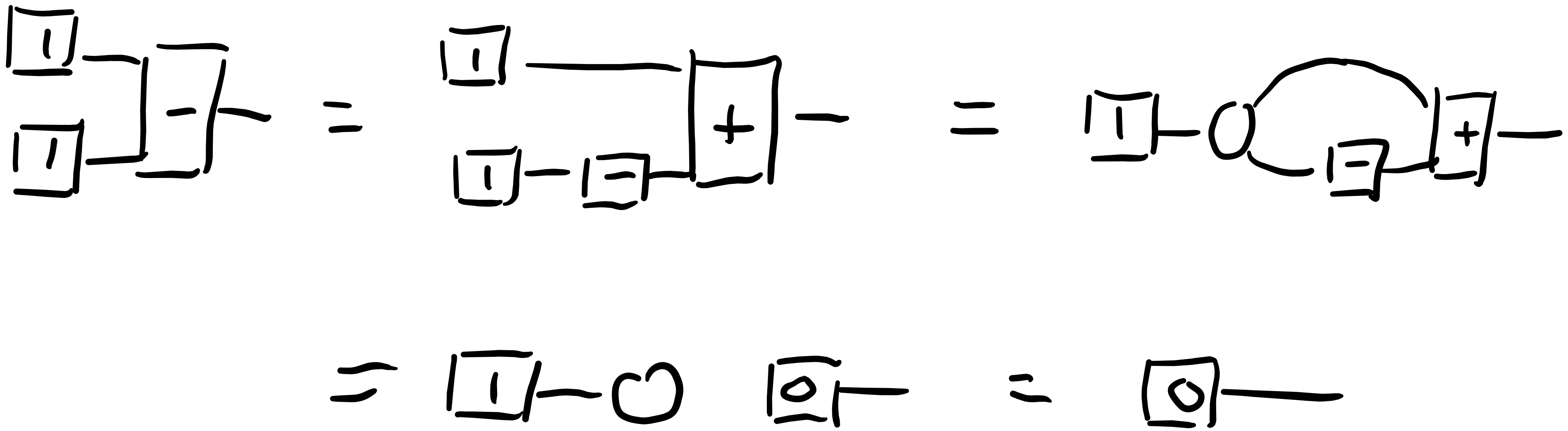
Exercise (5m)

Using cancellativity, prove that 1-1=0:



Exercise (Solutions)

Using cancellativity, prove that 1-1=0.



Binary Arithmetic with Pictures

Gates for binary arithmetic

In binary arithmetic, we only have the numbers 0 and 1. That is, we work with bits, or Boolean values, rather than integer numbers.

$0 \longleftrightarrow$ False (F)

$1 \longleftrightarrow$ True (T)

Gates for binary arithmetic

Multiplication is fine as it is (it’s the same as the logical conjunction AND).

×	0	1
0	0	0
1	0	1

AND	0 F	1 T
0 F	0 F	0 F
1 T	0 F	1 T

Gates for binary arithmetic

Addition, on the other hand, doesn't quite work as it is.

+	0	1
0	0	1
1	1	2

Not a valid binary value!

Gates for binary arithmetic

There are two possible ways to fix addition: making it addition modulo 2 (still written +) or making it logical disjunction (OR).

+	0	1
0	0	1
1	1	0

1+1=0

OR	0 F	1 T
0 F	0 F	1 T
1 T	1 T	1 T

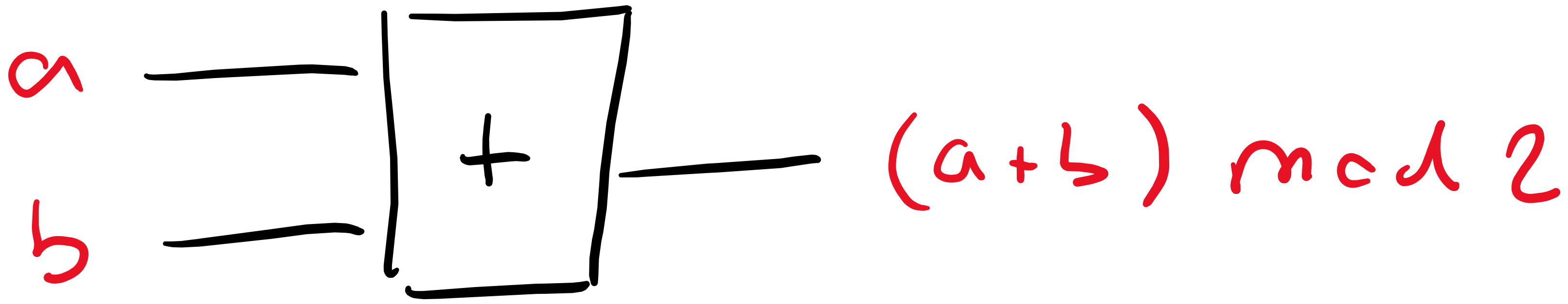
1 OR 1 = 1

T OR T = T

Sometimes also called XOR, for eXclusive OR.

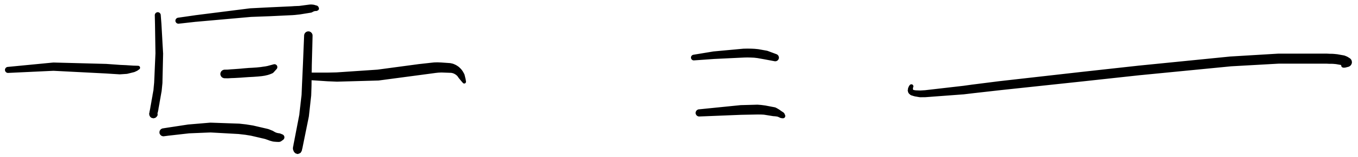
Gates for binary arithmetic

We define addition to be addition modulo 2. Logical disjunction can be defined from this addition and multiplication.



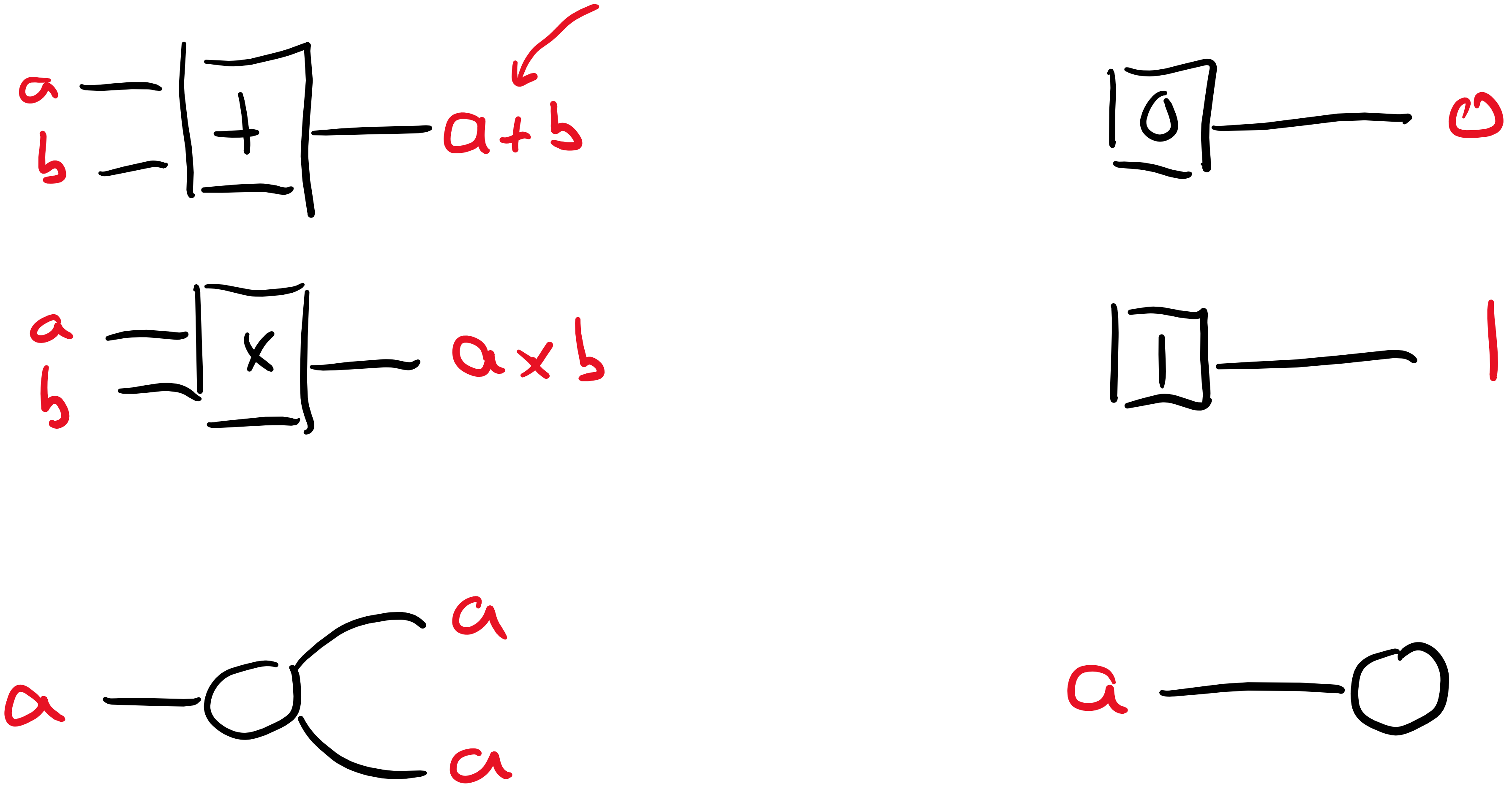
Gates for binary arithmetic

Negation is no longer necessary: we have that 1+1=0, so the negative of 1 is 1 itself.



Gates for binary arithmetic

(from now on, implicitly mod 2)



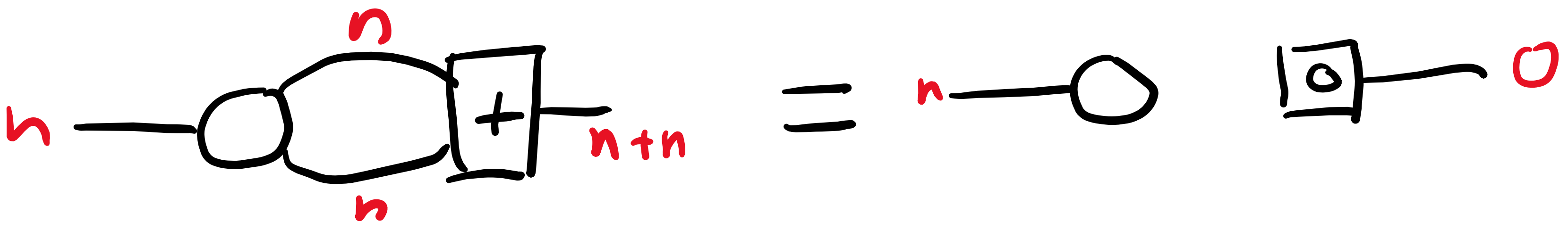
Rules for binary arithmetic

Pretty much the same as for ordinary arithmetic:

- Associativity of +, x and copy
- Commutativity of +, x and copy
- Neutral element for +, x and copy
- Distributivity of x on +
- Copy/delete rules for 0, 1, + and x

Cancellativity

Because there is no negation, cancellativity is simplified, becoming the statement that n + n = 0 for all n (i.e. for n=0, 1).

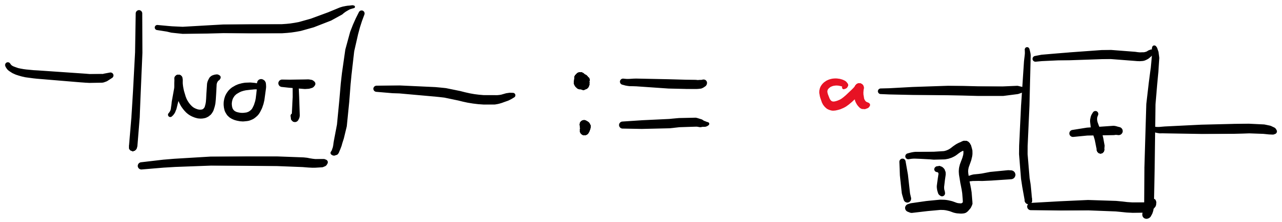


Exercise (5m)

Define the NOT gate using addition and the preparation of 1.

$$a \;-\!\boxed{\text{NOT}}\!-\; \begin{cases} 1 & \text{if } a=0 \\ 0 & \text{if } a=1 \end{cases}$$

Exercise (Solutions)

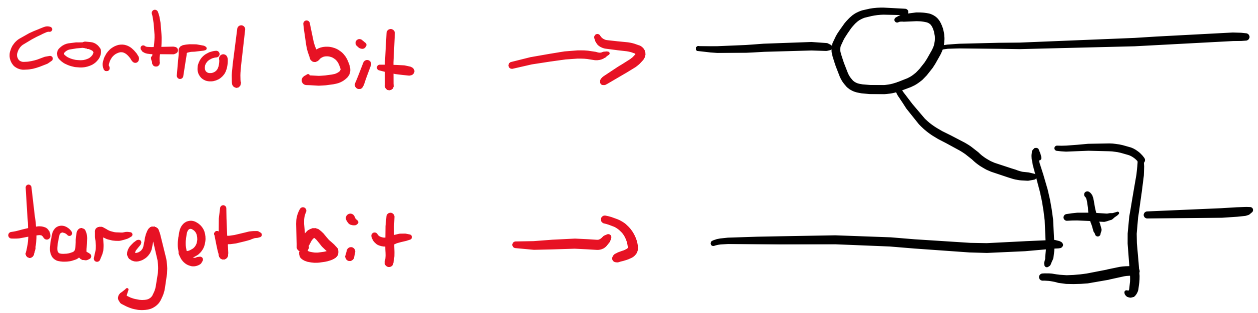


$$a+1 = \begin{cases} 1 & \text{if } a=0 \\ 0 & \text{if } a=1 \end{cases}$$

(because $1+1=0$)

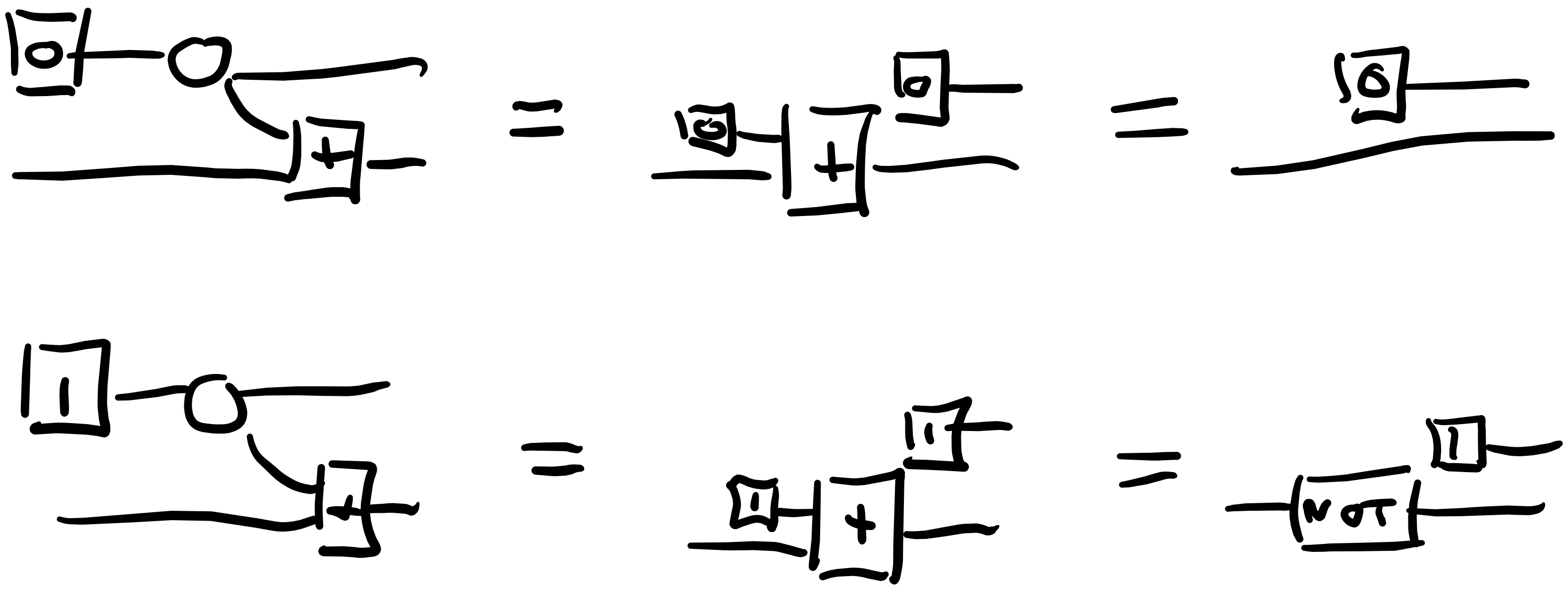
CNOT gate

The definition of the NOT gate in terms of addition inspires the following Controlled NOT gate, aka CNOT gate.



CNOT gate

When the control bit is 1, the target bit has a NOT applied to it.
When the control bit is 0, the target bit is unchanged.
The control bit is always copied through unchanged.

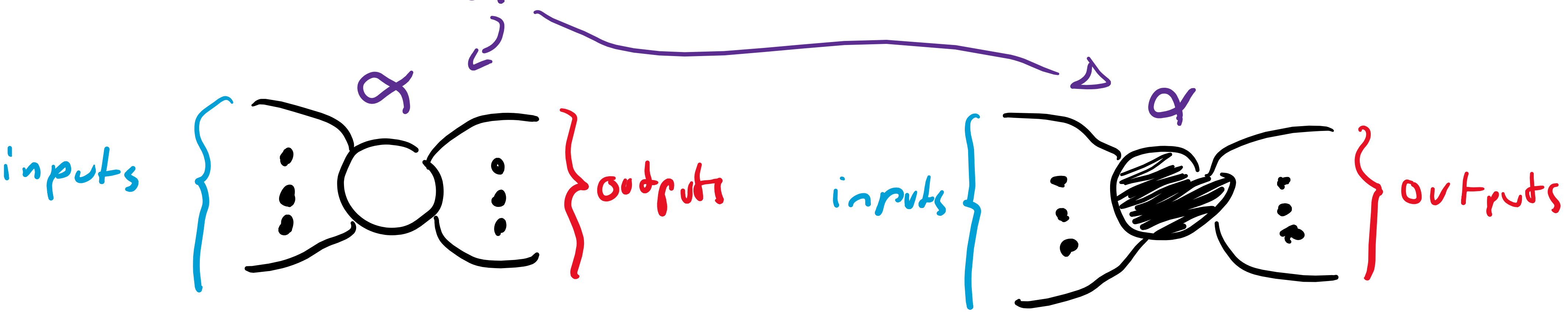


Let’s take a short break (15m)

Quantum Arithmetic with Pictures

Spiders

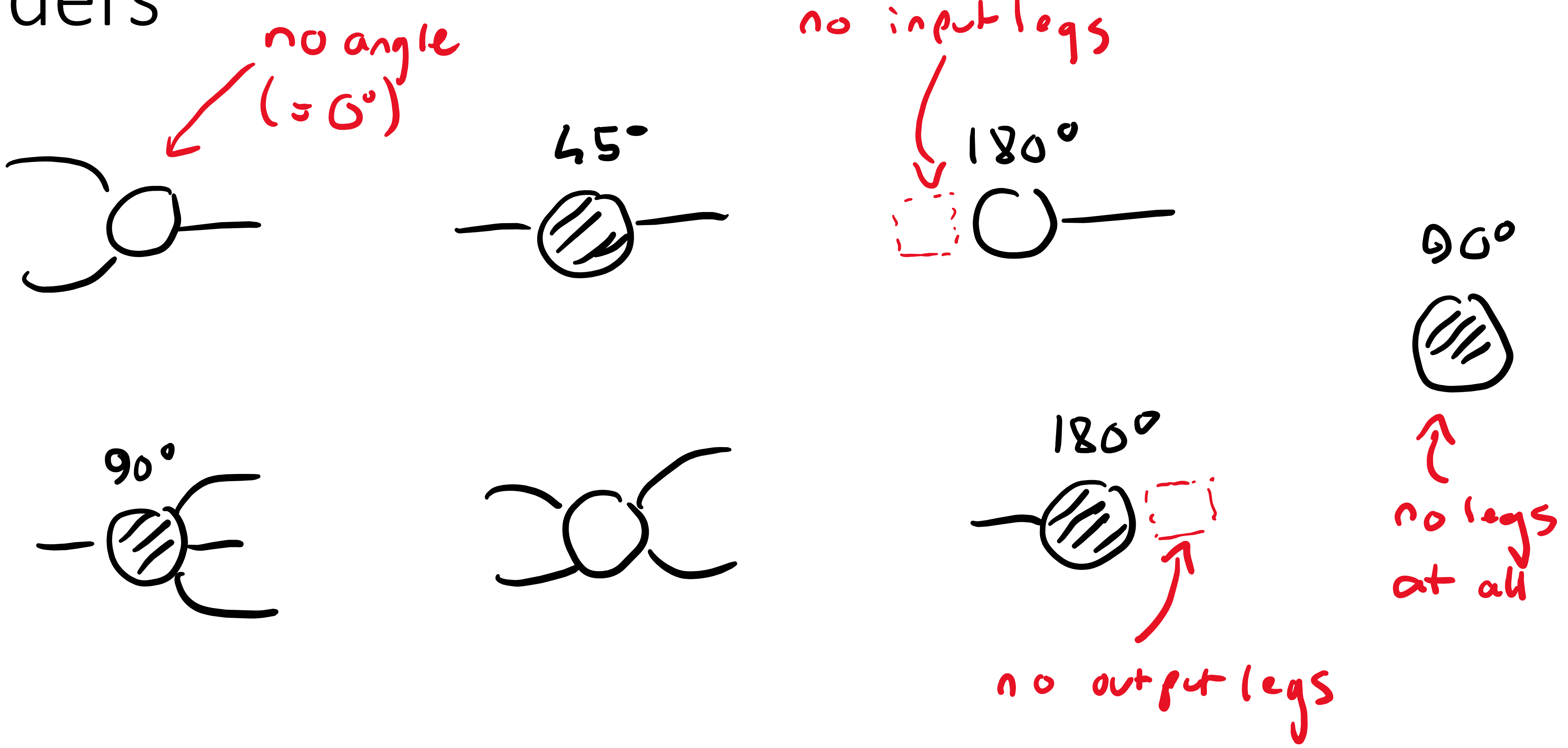
Much of quantum arithmetic is done with special gates known as "spiders". These come in two flavours: Z spiders and X spiders. They can have any number of inputs and any number of outputs. They have an angle associated (usually omitted if 0°).



Z spiders

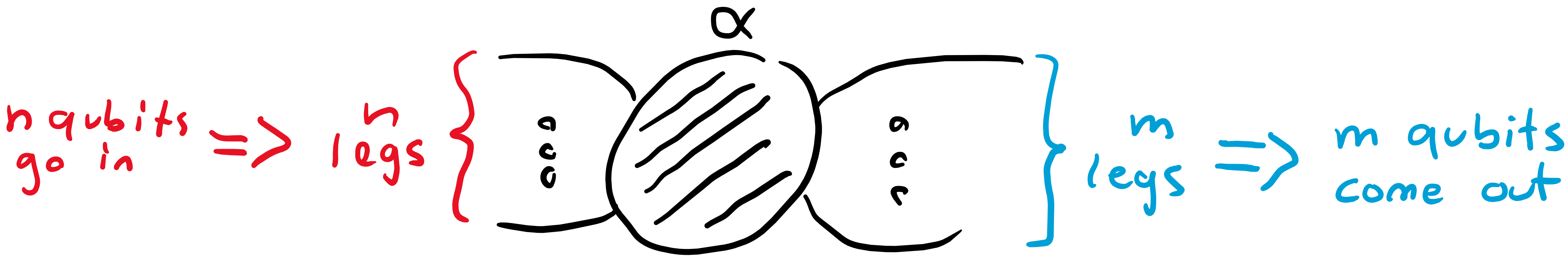
X spiders

Spiders


no angle
(= 0°)
45°
no input legs
180°
90°
90°
180°
no legs at all
no output legs

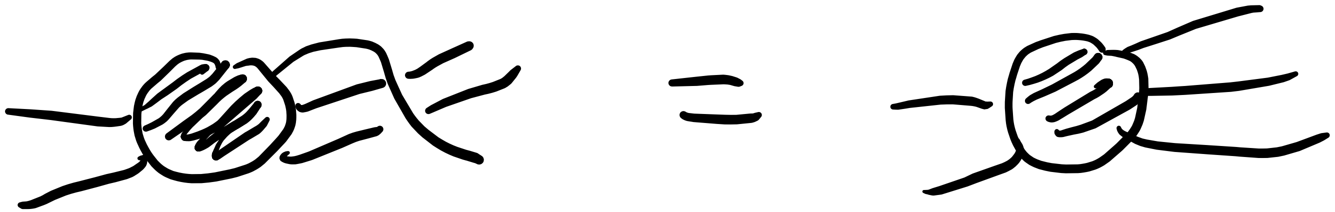
Spiders

Each leg of a spider carries one qubit. The generic spider below is a gate that takes n qubits in input and returns m qubits in output.



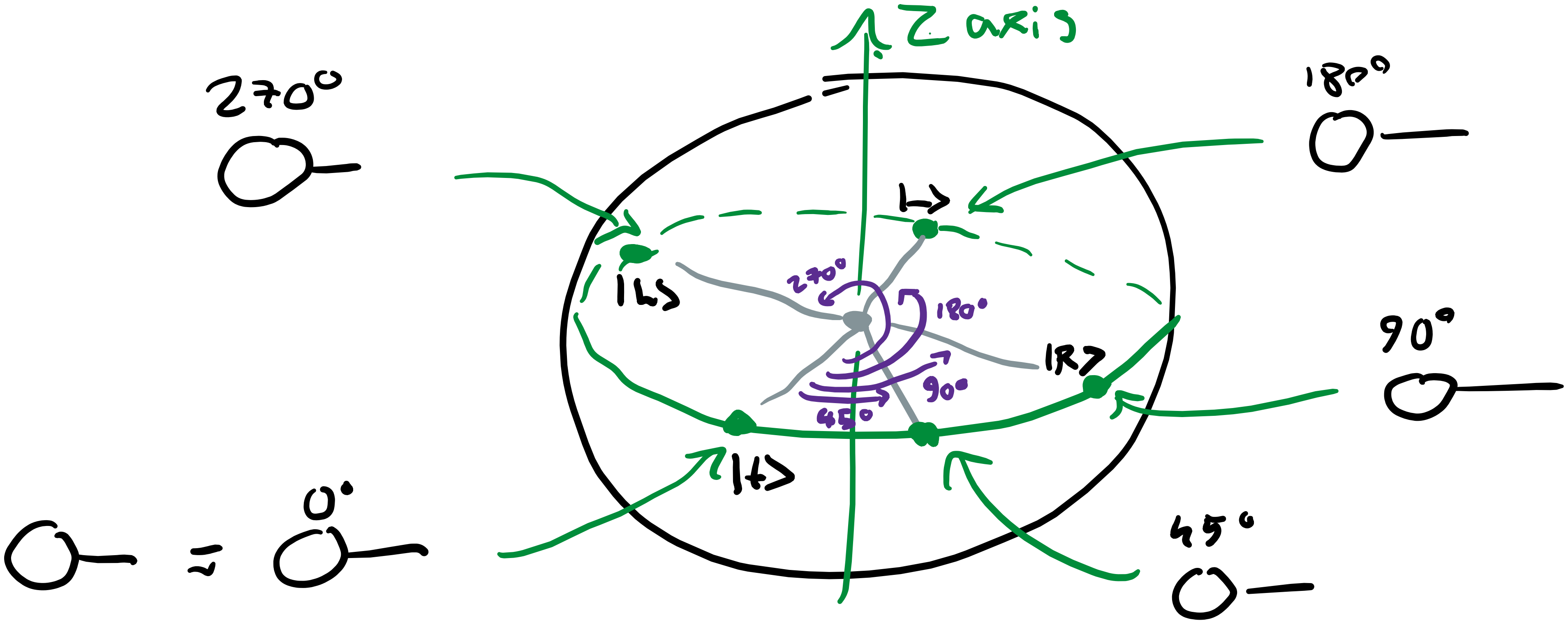
Commutativity for Spiders

The ordering of input (resp. output) legs in spiders is irrelevant.
Think of this as an extreme version of commutativity.



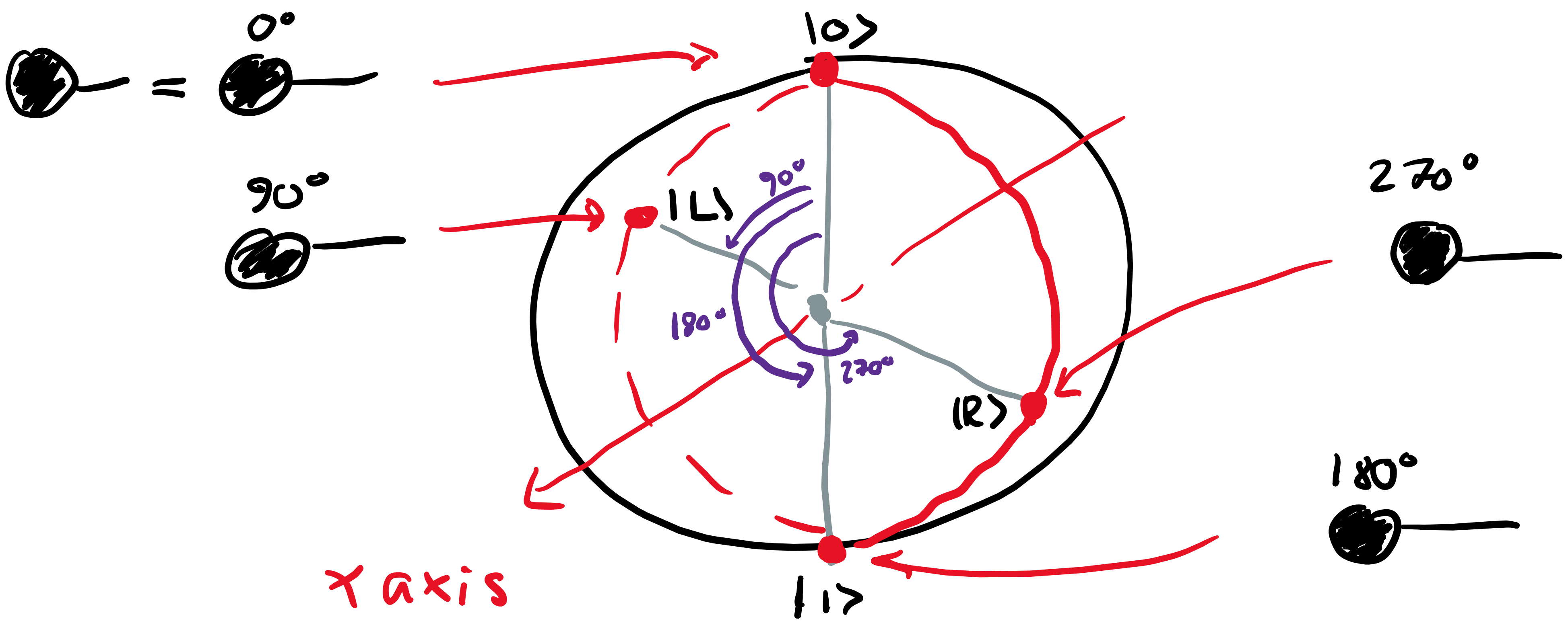
Spiders

Spider "states" are those with no input legs and a single output leg. These are the qubit states lying on the equator of the Bloch sphere (with respect to the Z and X axis respectively).



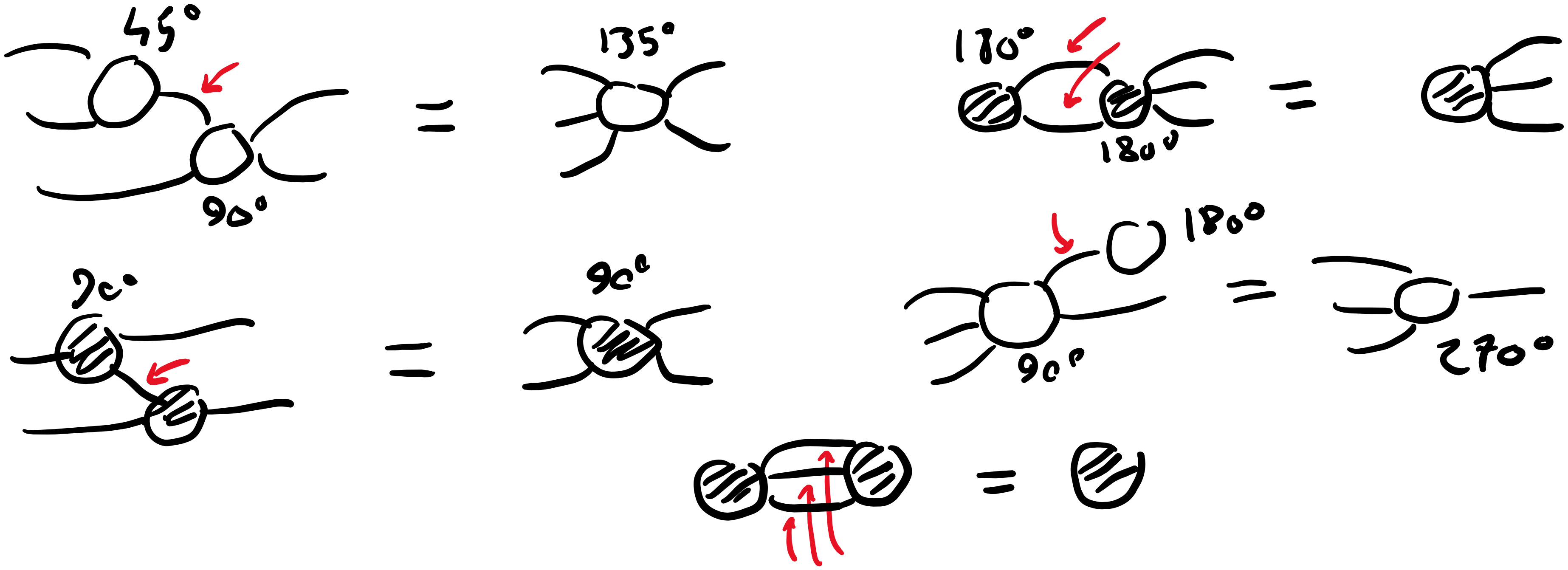
Spiders

Spider "states" are those with no input legs and a single output leg. These are the qubit states lying on the equator of the Bloch sphere (with respect to the Z and X axis respectively).



Spider fusion

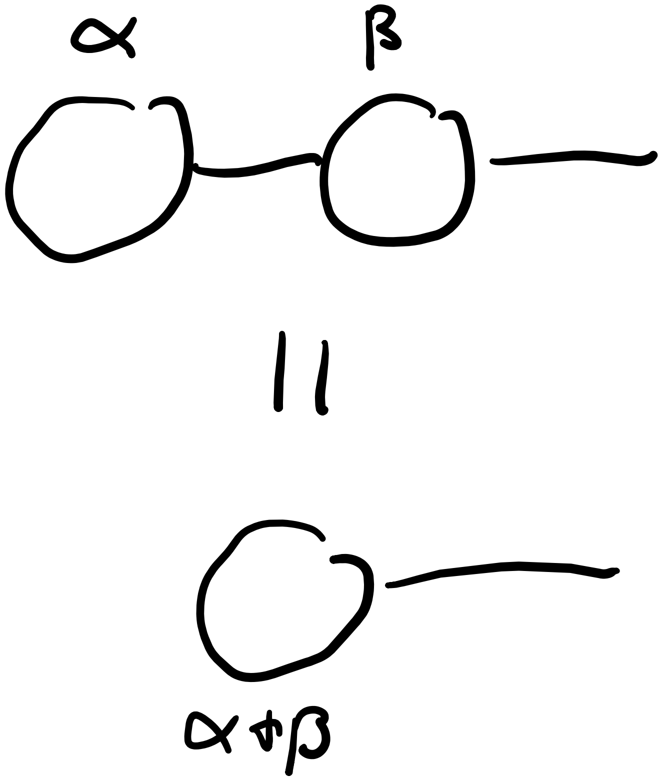
Spiders of the same colour sharing at least one leg can be "fused" together. The angles are added in the process.

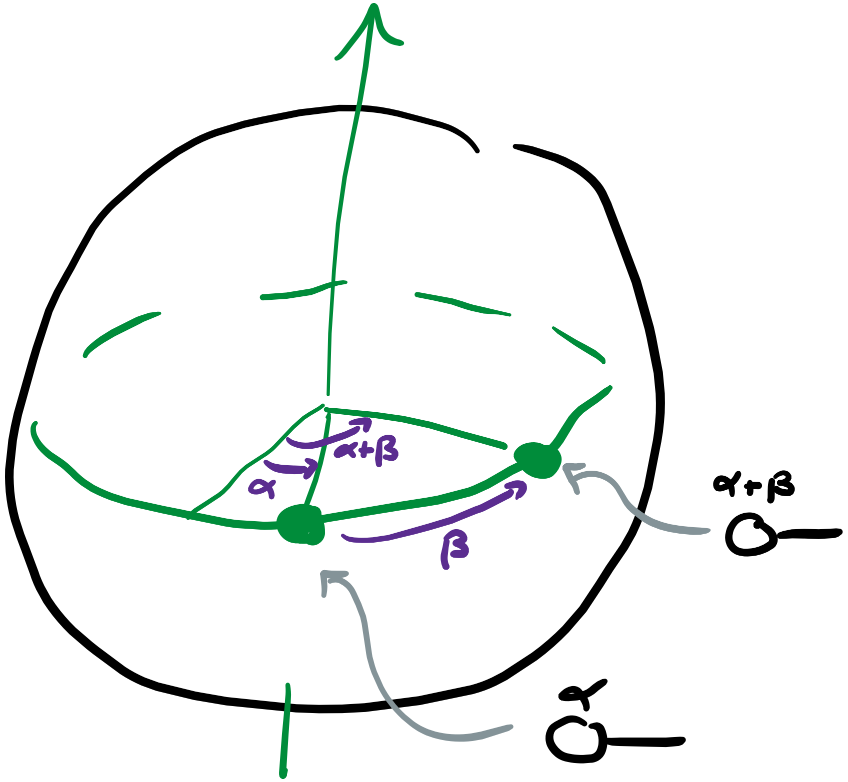


NB: Number of input/output legs is the same on both sides of the equation

Rotation spiders

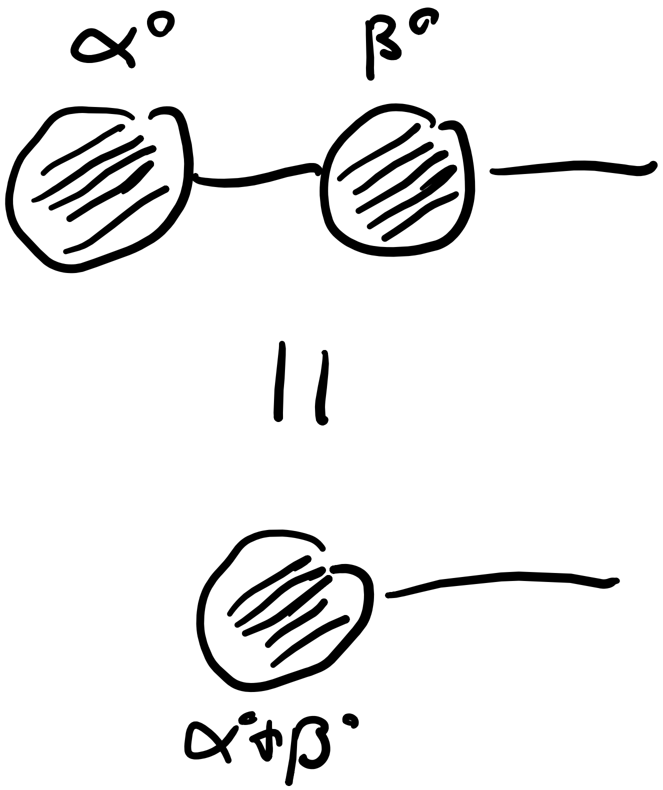
Spiders with 1 input leg and 1 output leg are the qubit rotations about the Z and X axis respectively, because of fusion:

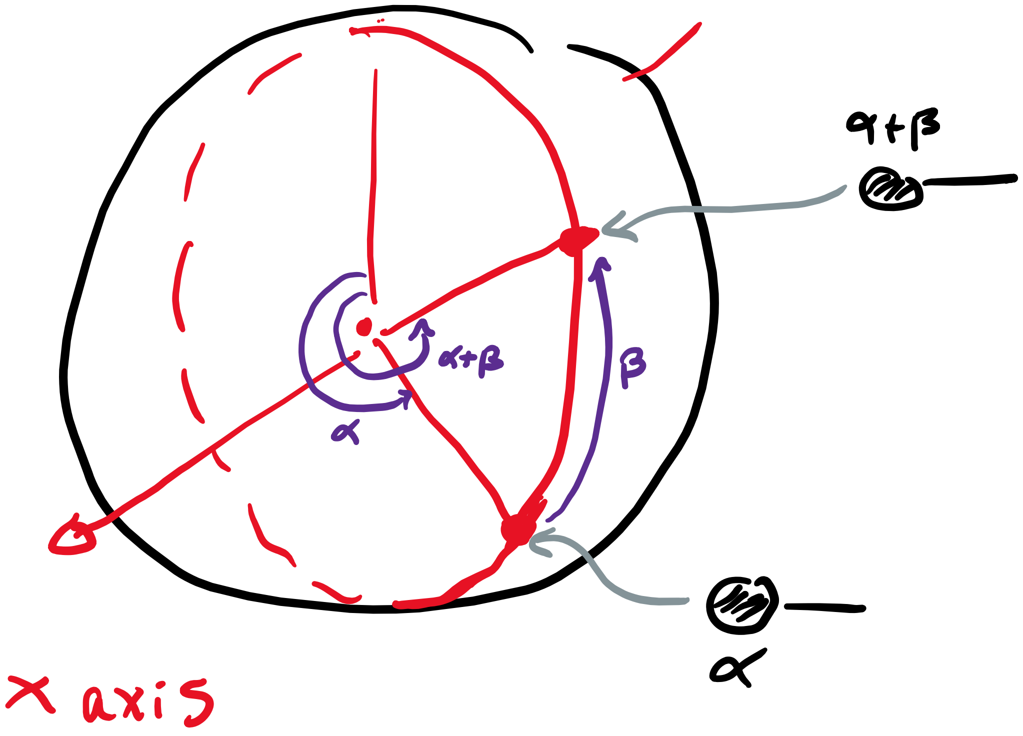




Rotation spiders

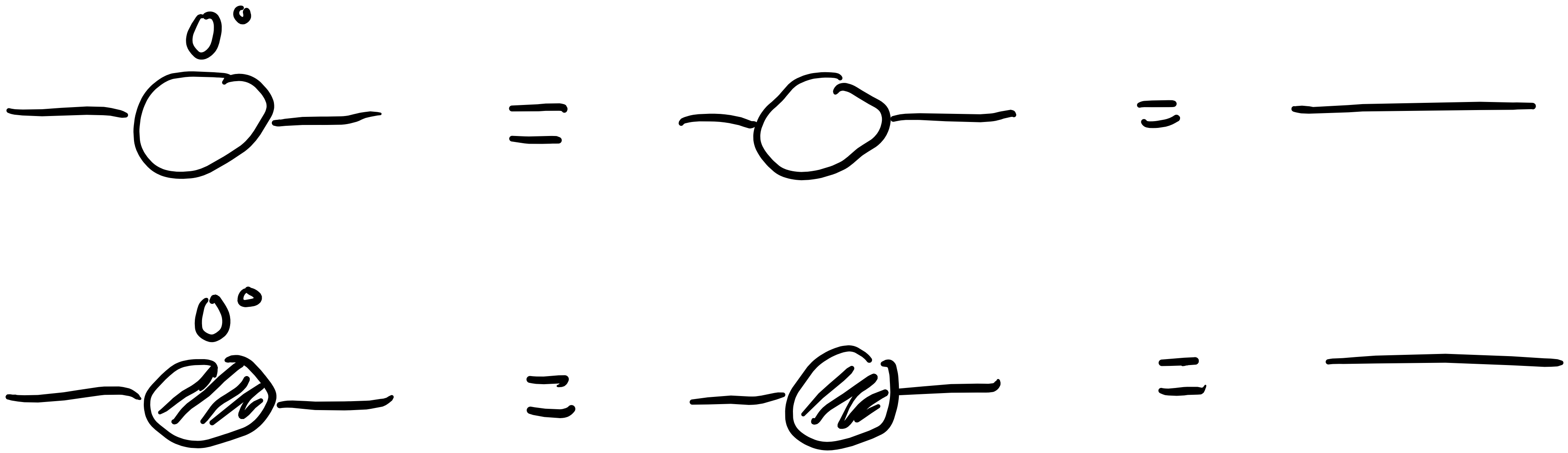
Spiders with 1 input leg and 1 output leg are the qubit rotations about the Z and X axis respectively, because of fusion:





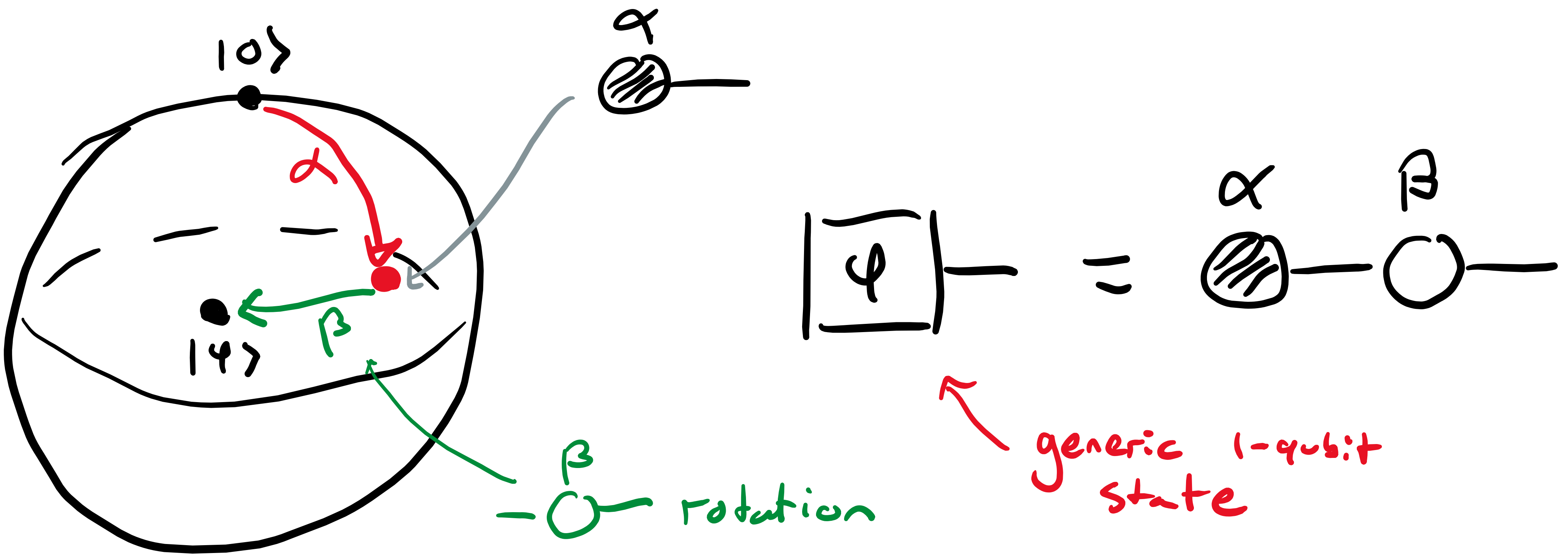
Identity spiders

Spiders with 1 input leg, 1 output leg and an angle of 0° are rotations by 0°: they do nothing, and hence can be omitted.



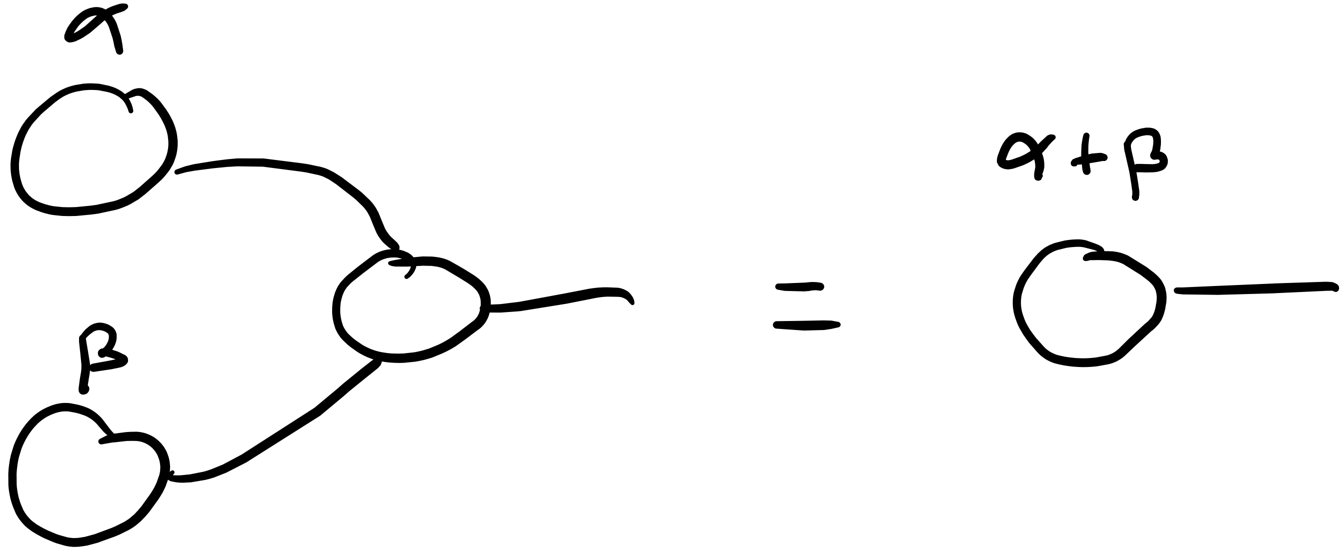
All qubit states from spiders

All qubit states can be obtained by using two spiders, e.g. an X spider state (selecting the latitude on the sphere) followed by a Z rotation (selecting the longitude on the sphere):



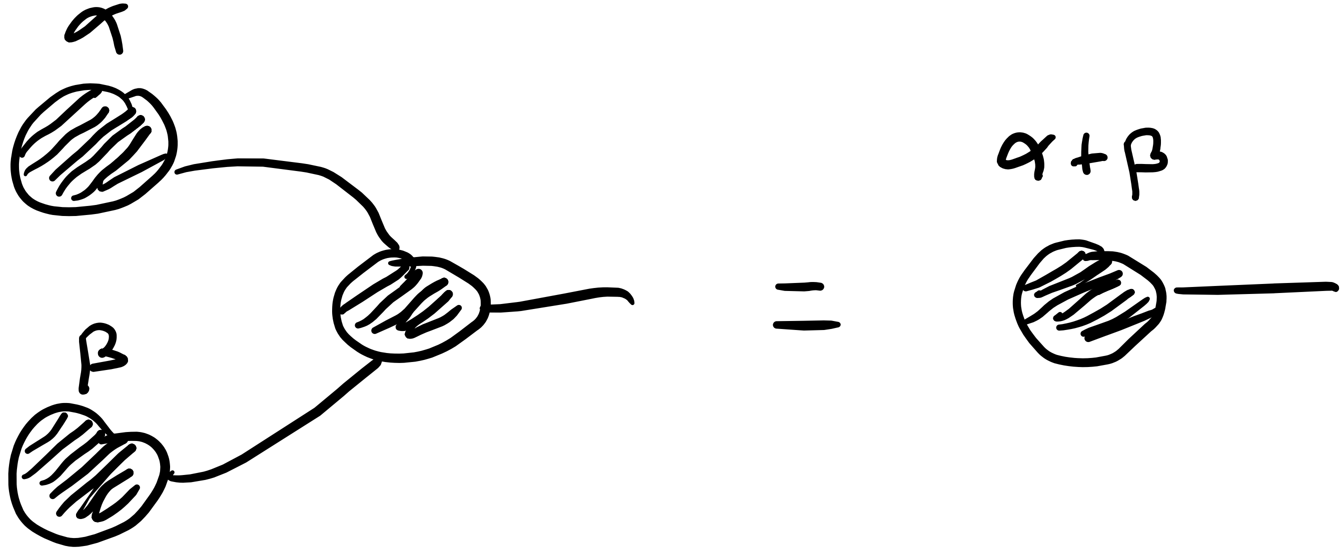
Addition spiders

Spiders with 2 input legs, 1 output leg and an angle of 0° act as "addition spiders": they take two spider states and return a spider state with the sum of their angles.



Addition spiders

Spiders with 2 input legs, 1 output leg and an angle of 0° act as "addition spiders": they take two spider states and return a spider state with the sum of their angles.

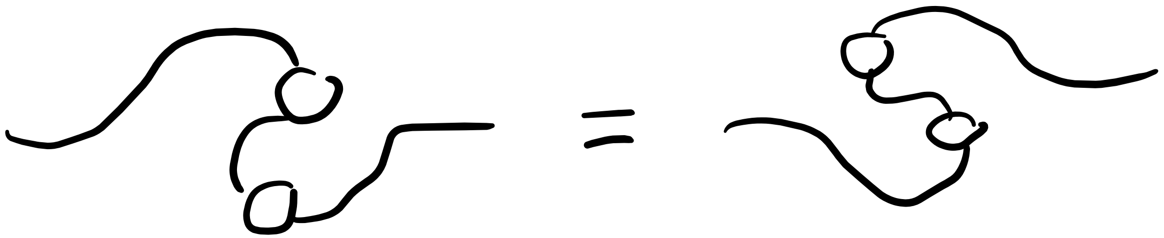


Exercise (5m)

Prove the following properties using spider fusion:

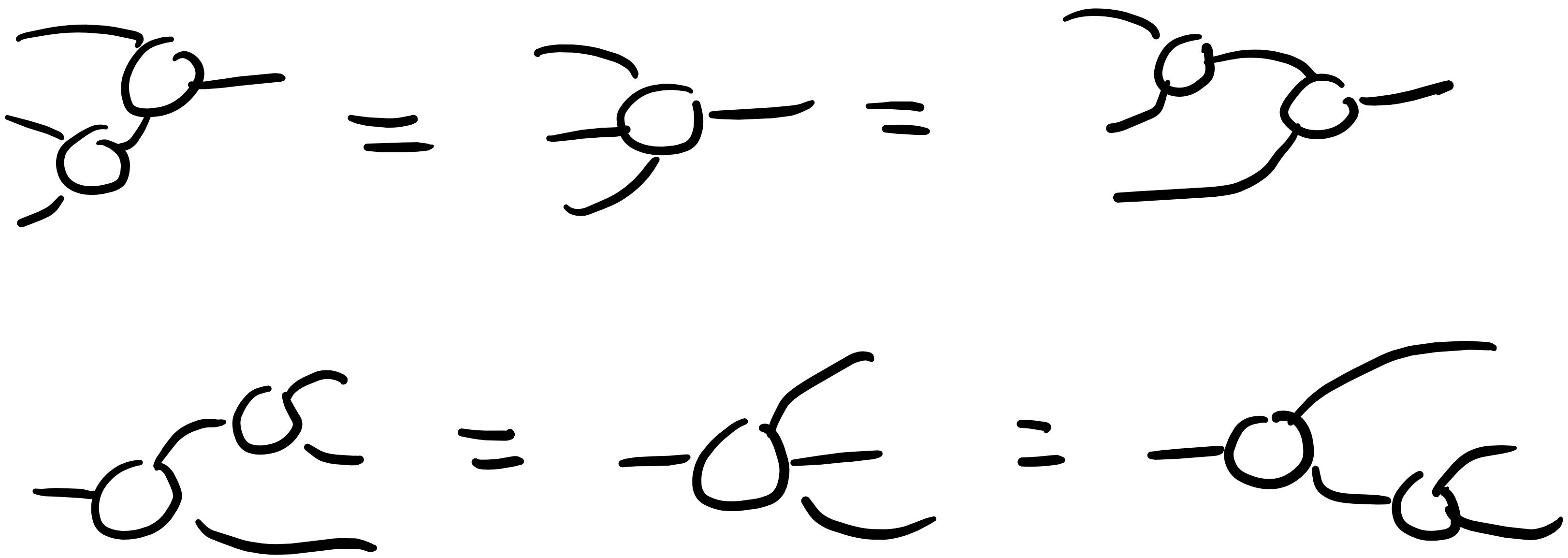
- Associativity

- Snake equations

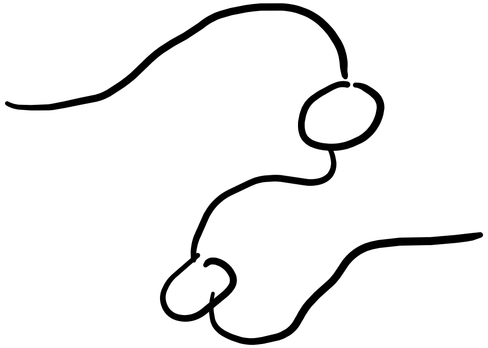


- Neutral element

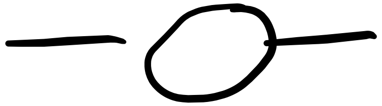
Exercise (Solutions)



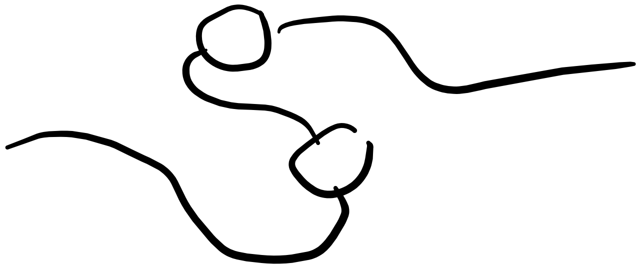
Exercise (Solutions)



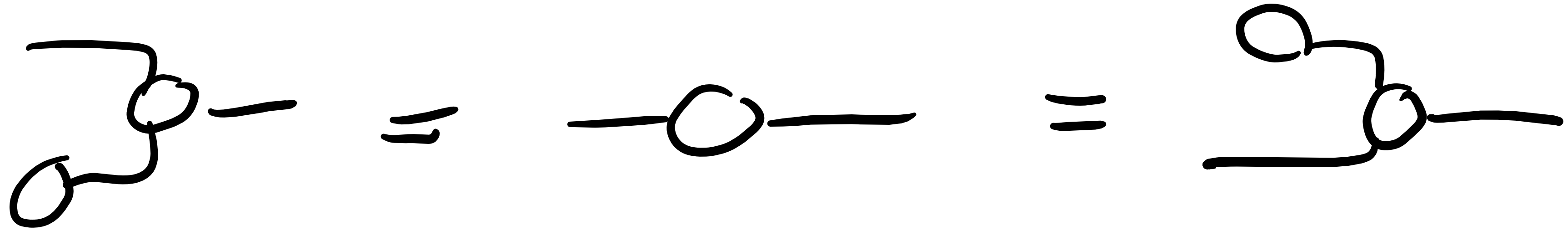
=

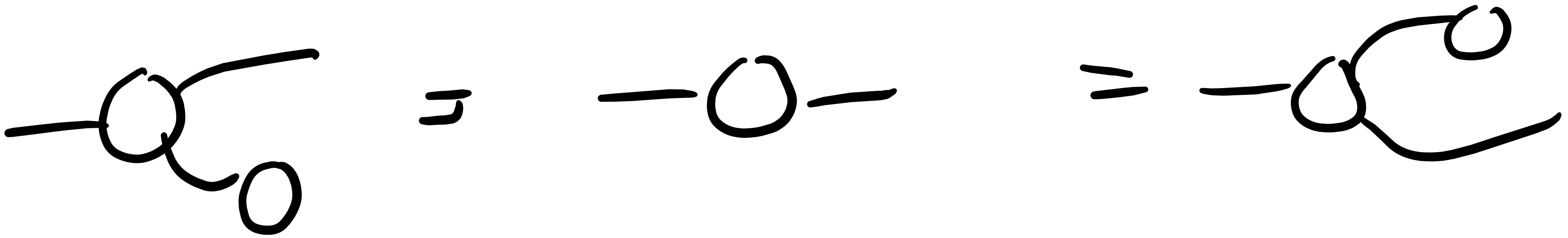


=



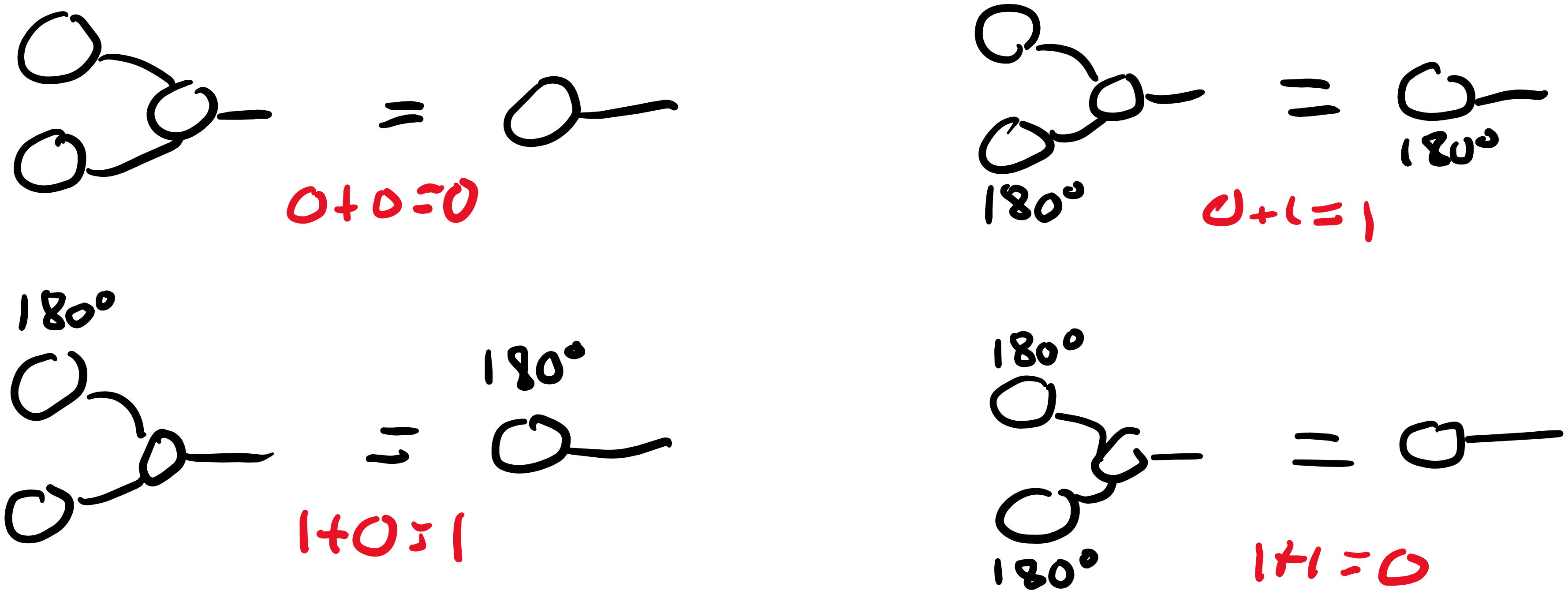
Exercise (Solutions)





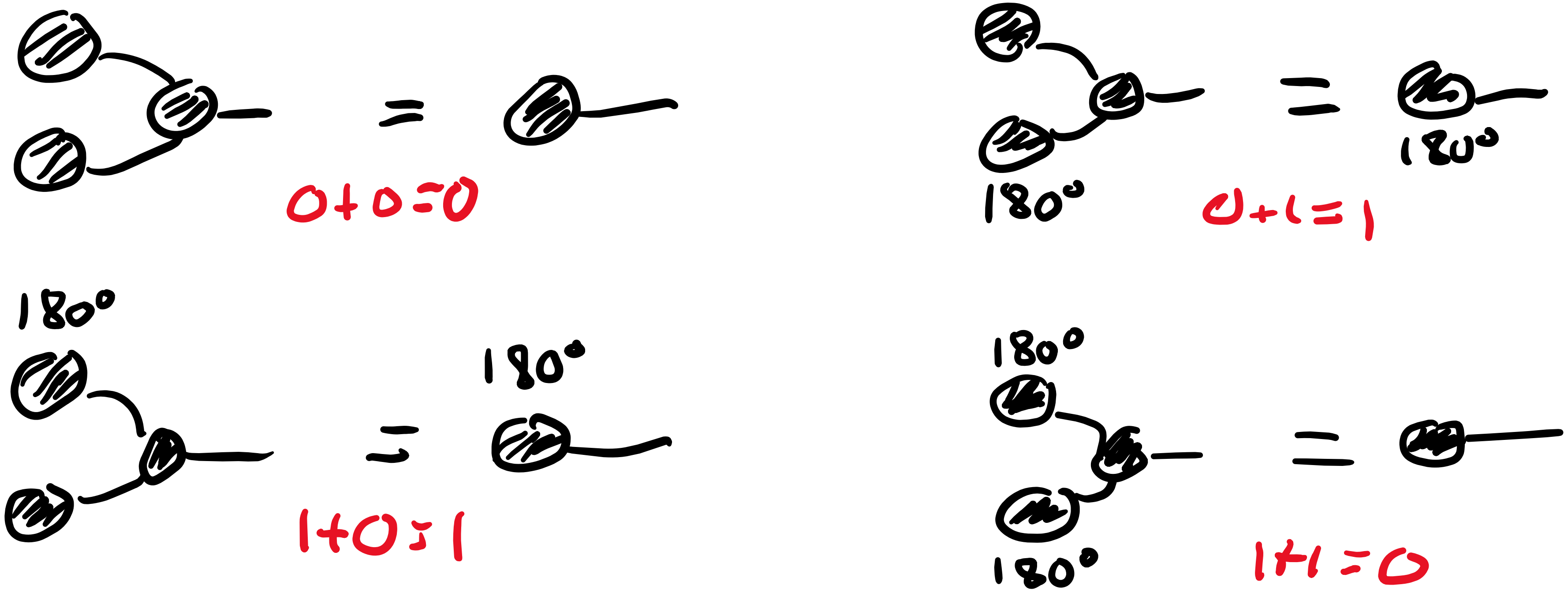
Special spider states

Because 180°+180° = 360° = 0°, the spider states with 0° and 180° in each colour can be used to encode a bit, with the corresponding addition spider acting as binary addition.



Special spider states

Because 180°+180° = 360° = 0°, the spider states with 0° and 180° in each colour can be used to encode a bit, with the corresponding addition spider acting as binary addition.

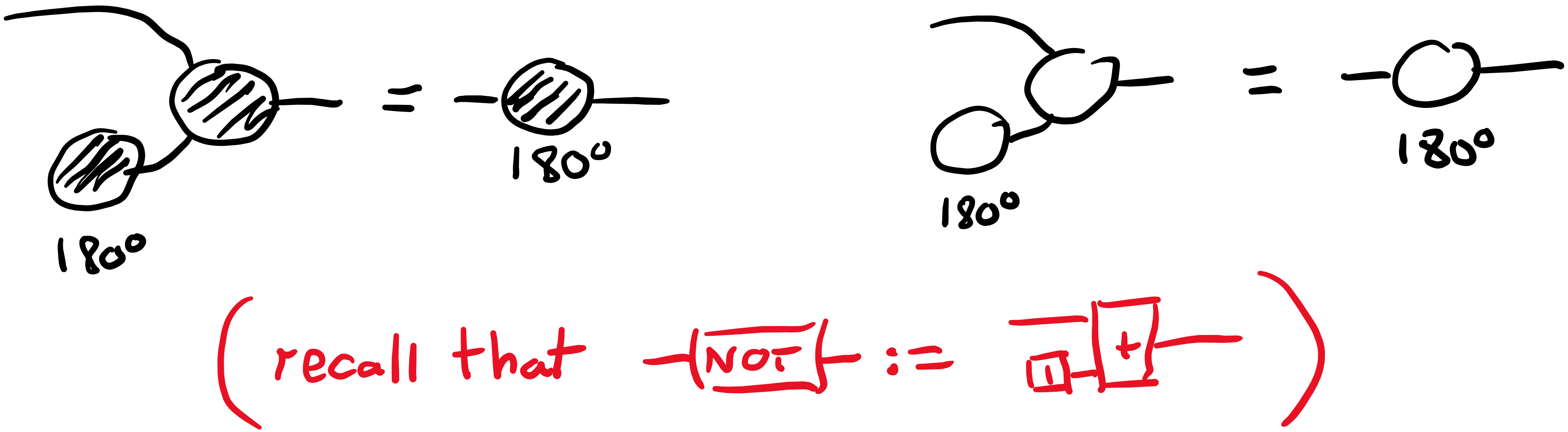


Special spider states

Because 180°+180° = 360° = 0°, the spider states with 0° and 180° in each colour can be used to encode a bit, with the corresponding addition spider acting as binary addition.

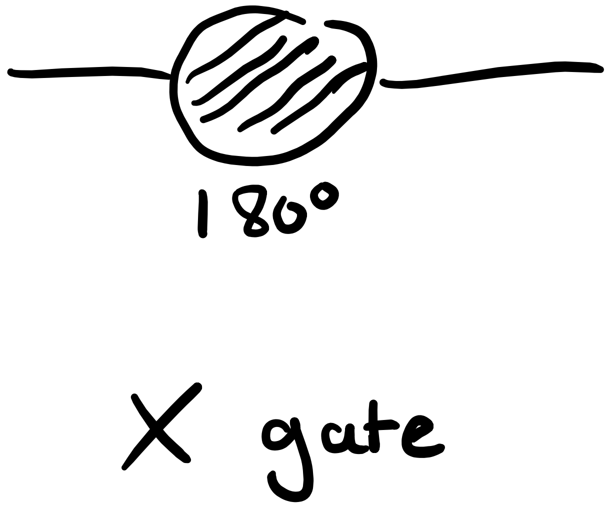
NOT gates for spider states

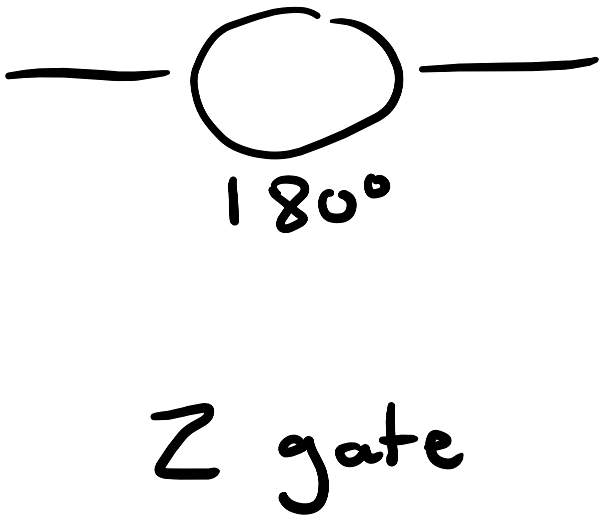
We have addition spiders and we have 180° spider states that act as the bit value 1: if we put them together, we get NOT gates for spiders (one per colour).



The X and Z gates

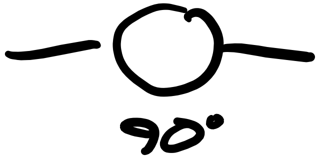
Because there are two possible NOT gates, on qubits, they get special names: they are called X gates and Z gates.





Other named rotations

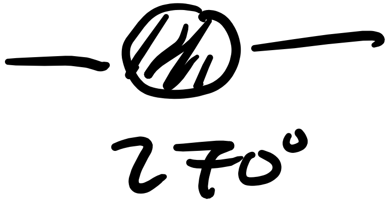
S gate



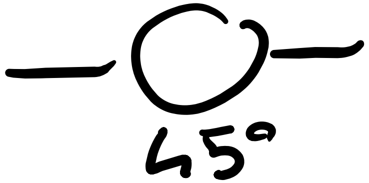
$S^\dagger$ gate

$\sqrt{x}$ gate

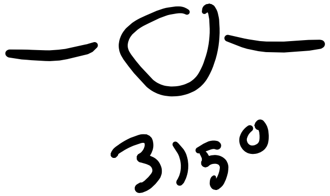
$\sqrt{x}^\dagger$ gate



T gate

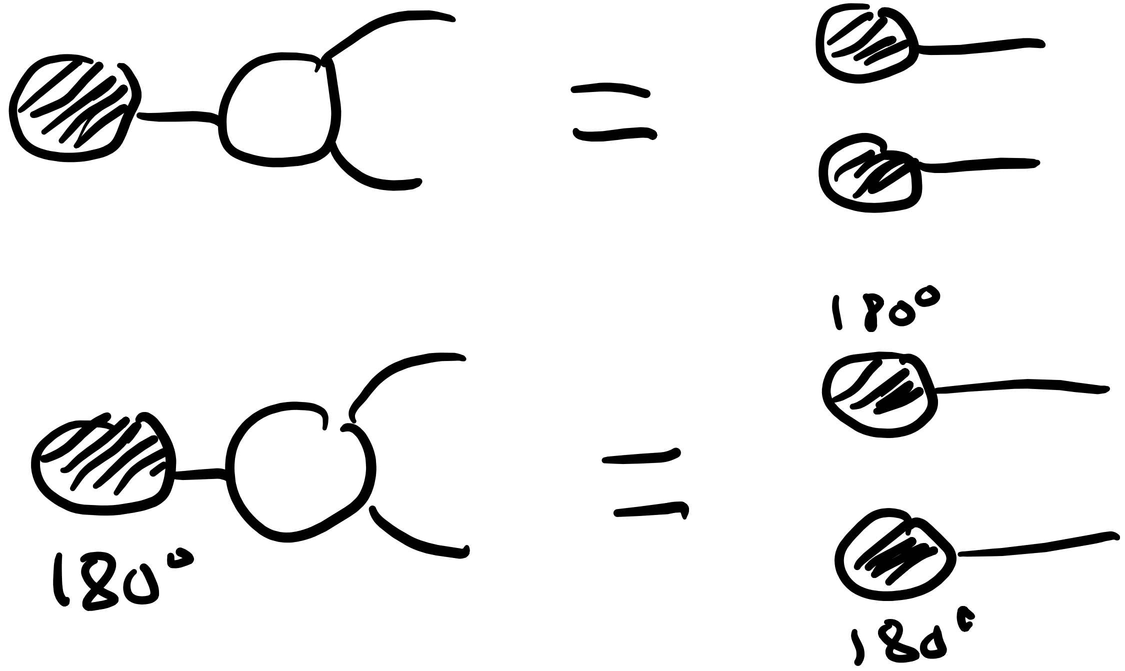


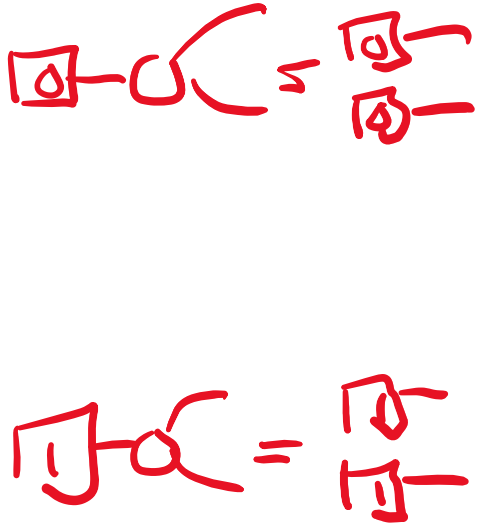
$T^\dagger$ gate



Copy rules for spider states

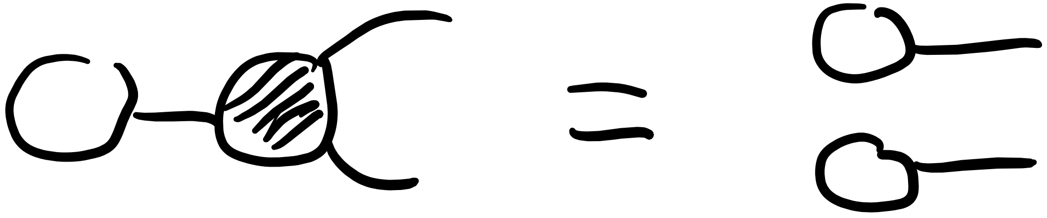
Spiders with 1 input leg, 2 output legs and an angle of 0° act as "copy spiders", but only on two states each (the spider states with angles 0° and 180° of the opposite colour).

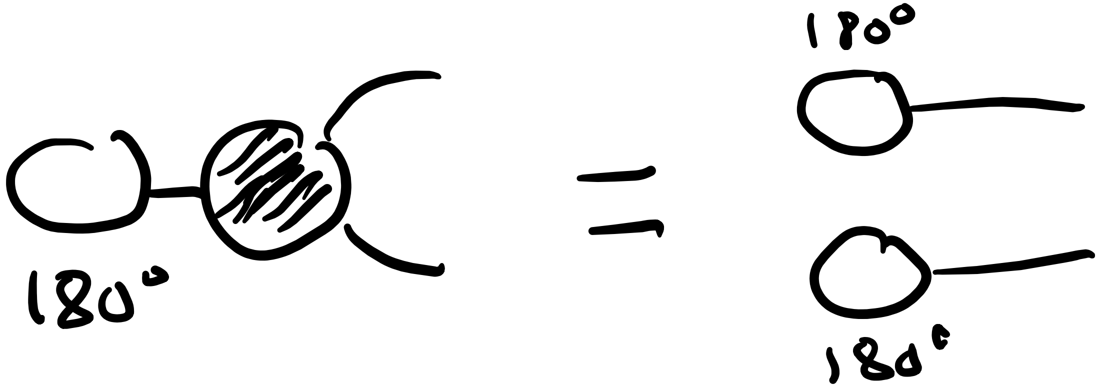




Copy rules for spider states

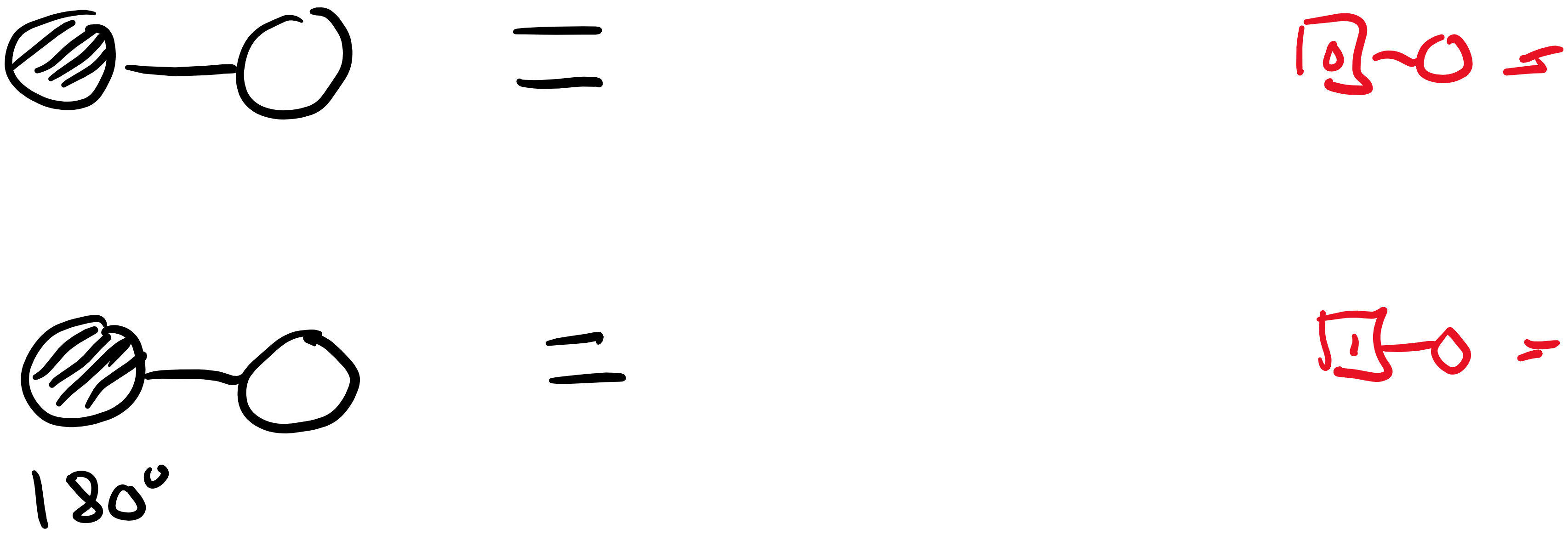
Spiders with 1 input leg, 2 output legs and an angle of 0° act as "copy spiders", but only on two states each (the spider states with angles 0° and 180° of the opposite colour).





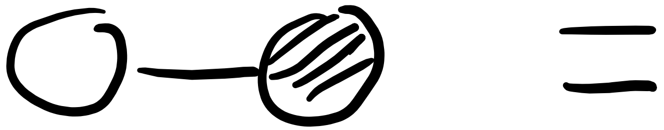
Delete rules for spider states

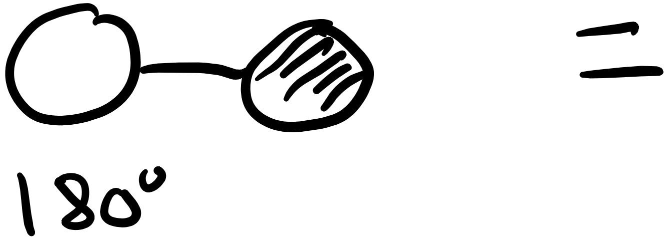
Spiders with 1 input leg, no output legs and an angle of 0° act as "delete spiders", but only on two states each (the spider states with angles 0° and 180° of the opposite colour).



Delete rules for spider states

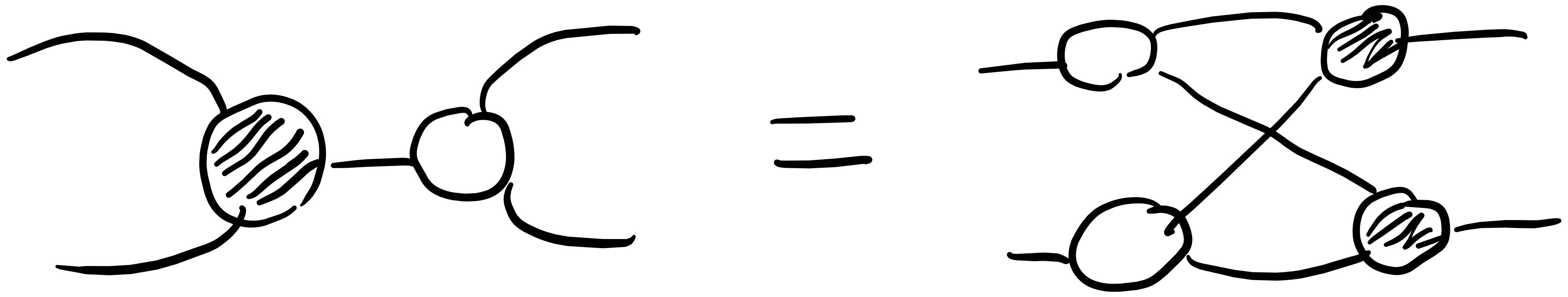
Spiders with 1 input leg, no output legs and an angle of 0° act as “delete spiders”, but only on two states each (the spider states with angles 0° and 180° of the opposite colour).





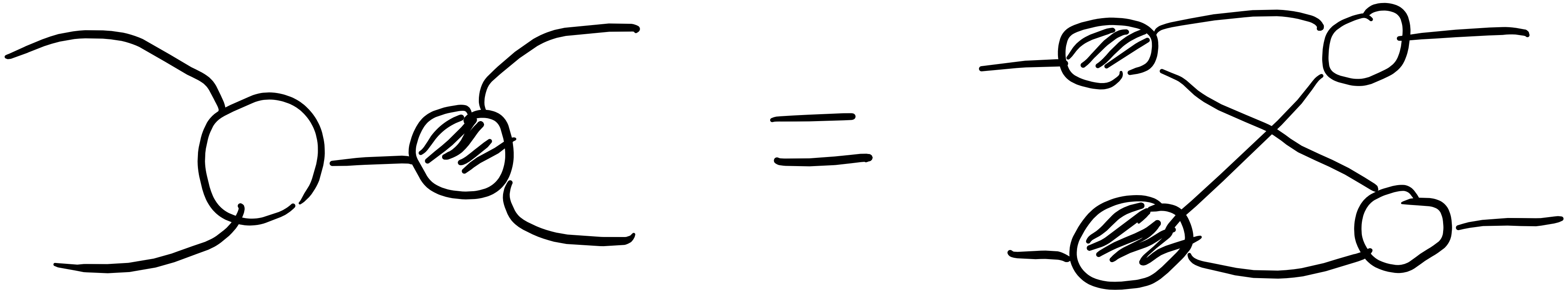
Copy rules for addition spiders

The copy rules for spiders relate the copy spider of one colour to the addition spider of the other.



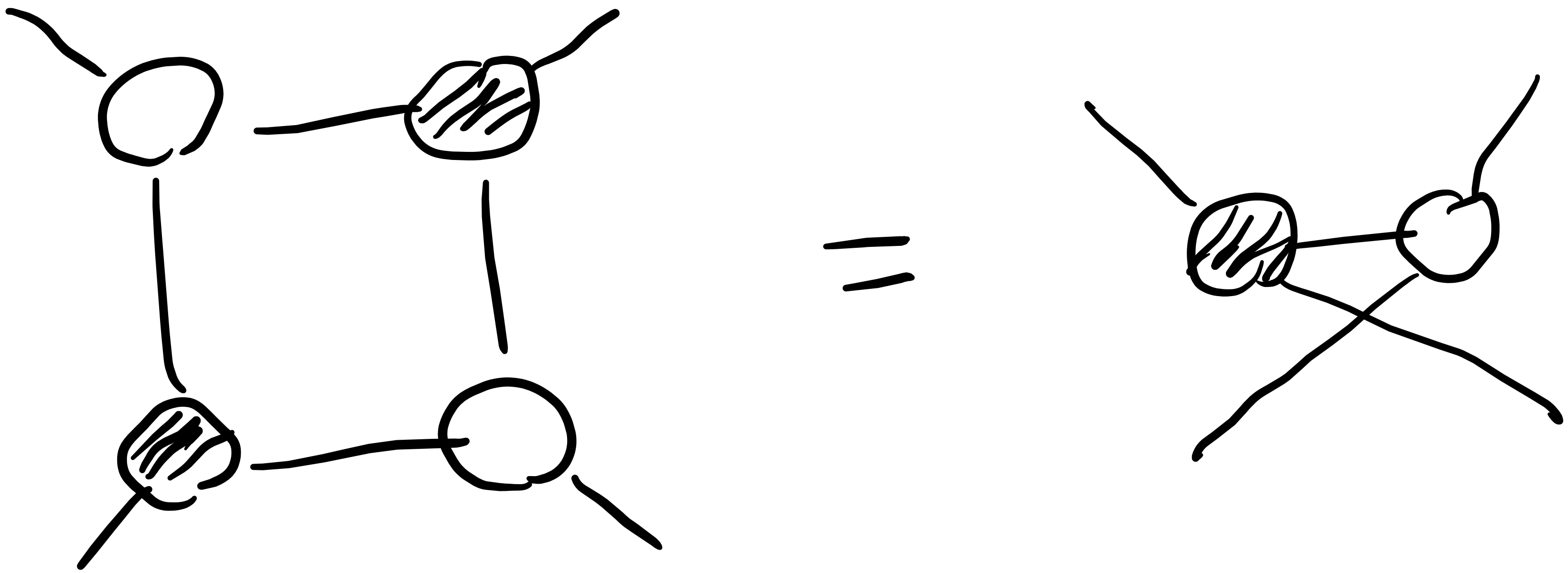
Copy rules for addition spiders

The copy rules for spiders relate the copy spider of one colour to the addition spider of the other.



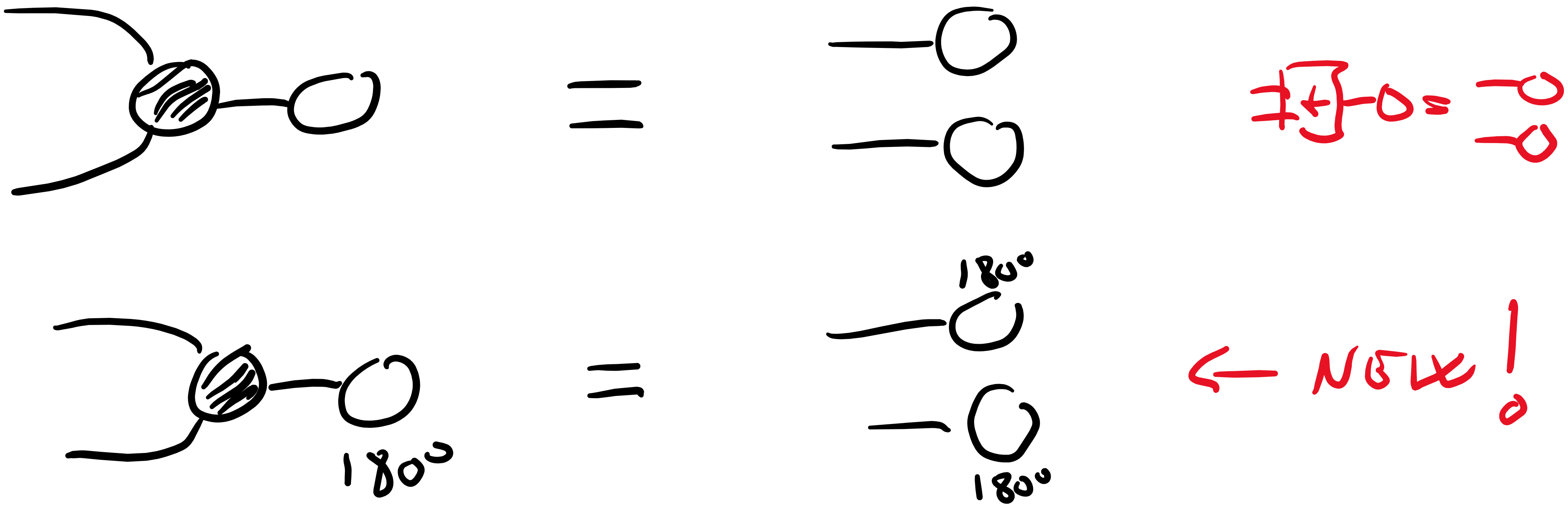
The square popping rule

An alternative way to see the copy rule for spider is the following "square-popping rule":



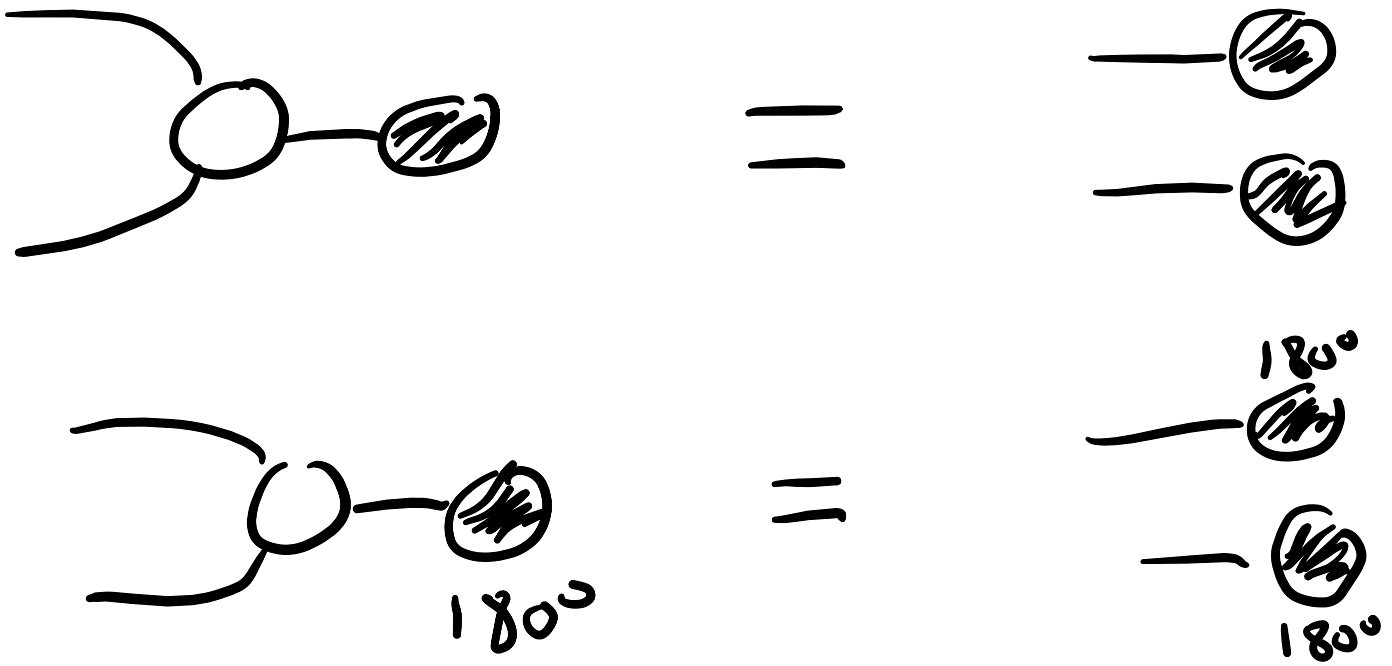
Delete rules for addition spiders

The delete rules for spiders relate the delete spider of one colour to the addition spider of the other.



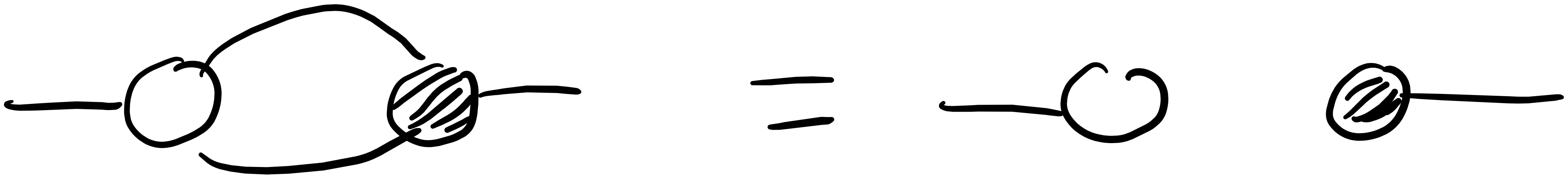
Delete rules for addition spiders

The delete rules for spiders relate the delete spider of one colour to the addition spider of the other.



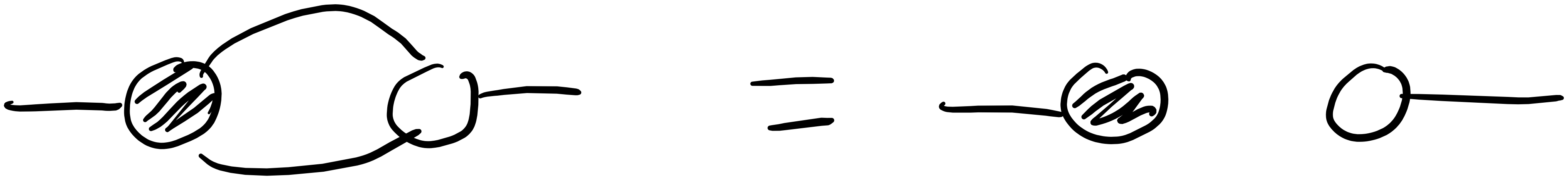
Cancellativity for spiders

There are two cancellativity rules for spiders, one per copy-addition pair of opposite colours.



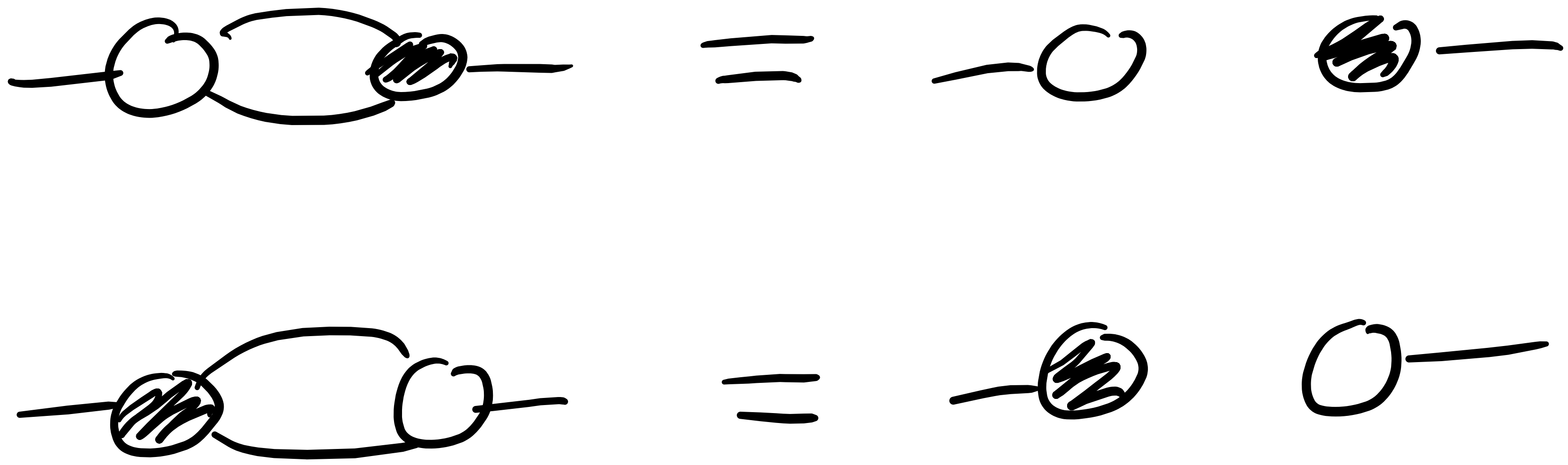
Cancellativity for spiders

There are two cancellativity rules for spiders, one per copy-addition pair of opposite colours.

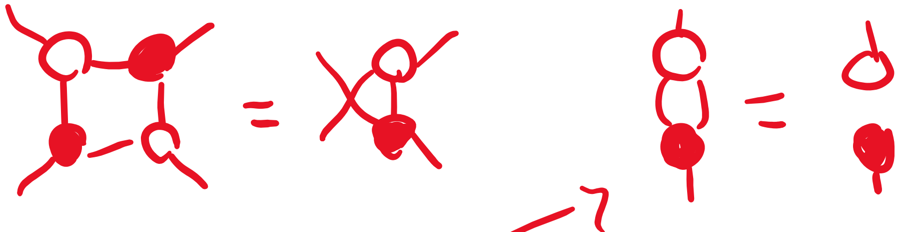


Leg-chopping rule

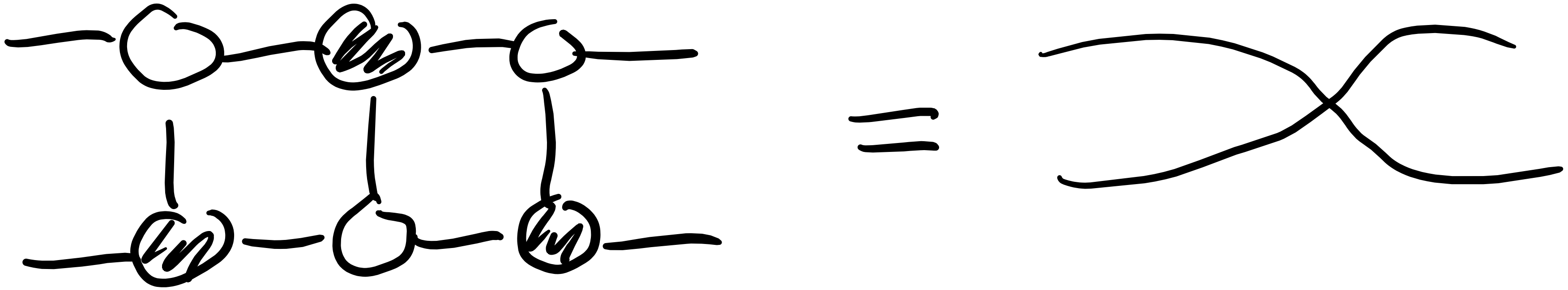
Cancellativity is also known as the “leg-chopping rule”.



Exercise (10m)

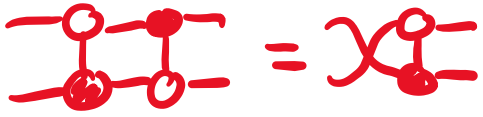


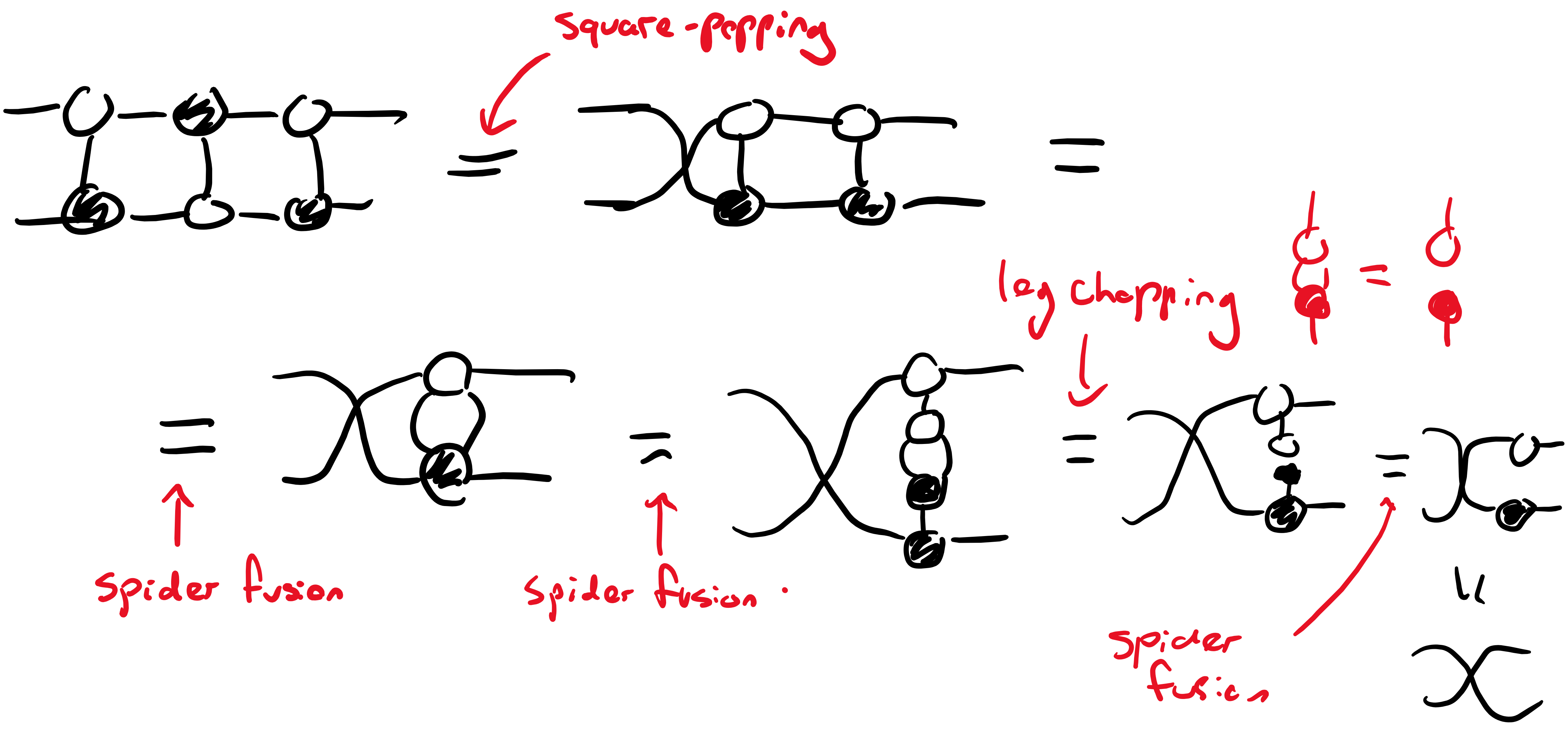
Use the square-popping, leg-chopping and spider fusion to prove the following equation:



(Later on, we'll see that the gates above are CNOTs. This equation then states that three alternating CNOTs can be used to swap qubits, a useful fact that finds many applications in quantum computing.)

Exercise (Solutions)

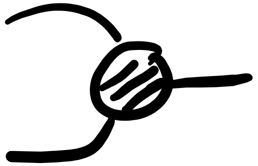




Spider arithmetic

At this point, we have two ways of doing binary arithmetic with spiders: one where the 0°/180° Z spider states play the role of a bit, another where the 0°/180° X spider states play the same.

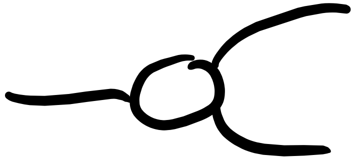
addition



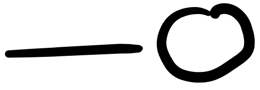
Prepare 0

Prepare 1

copy



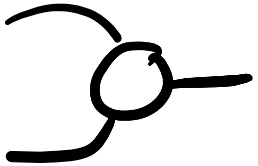
delete



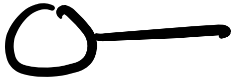
Spider arithmetic

At this point, we have two ways of doing binary arithmetic with spiders: one where the 0°/180° Z spider states play the role of a bit, another where the 0°/180° X spider states play the same.

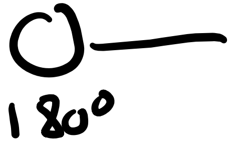
addition



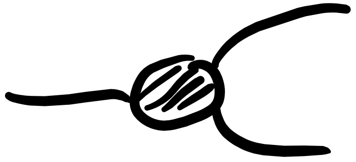
Prepare 0



Prepare 1



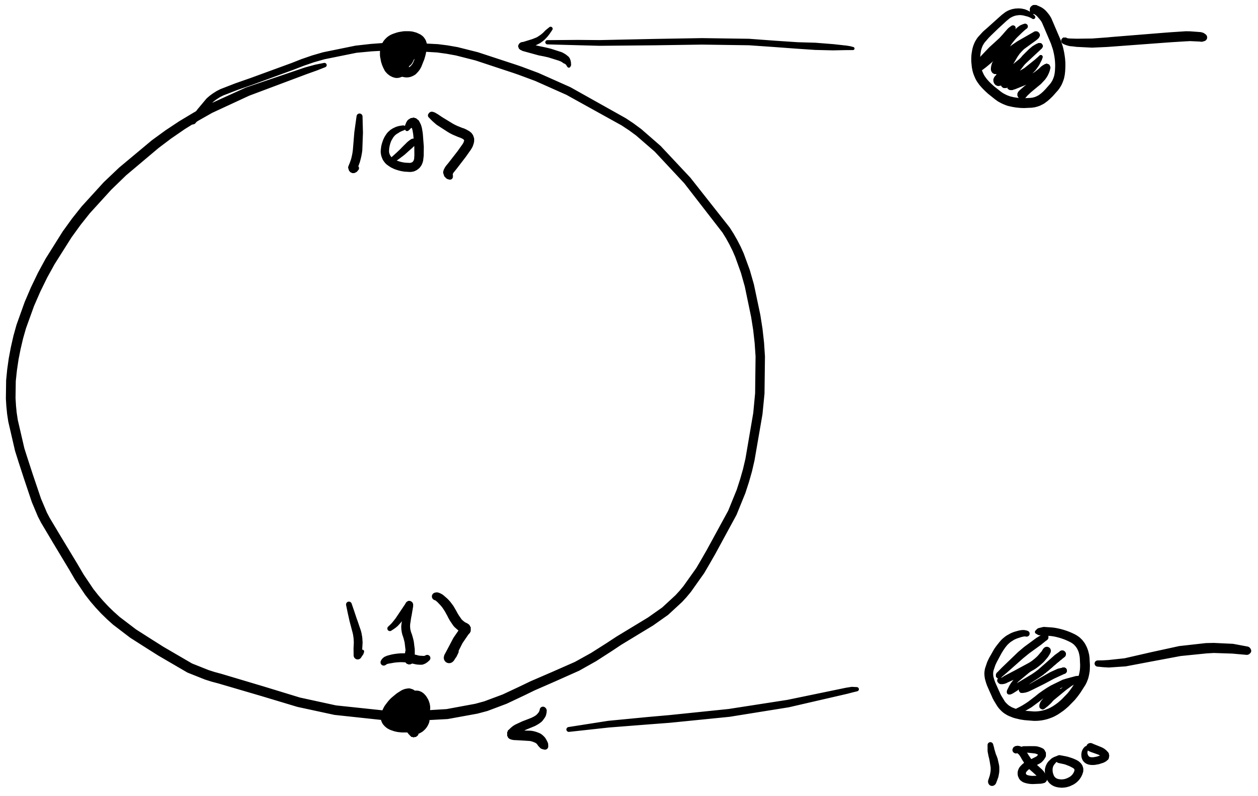
copy



delete

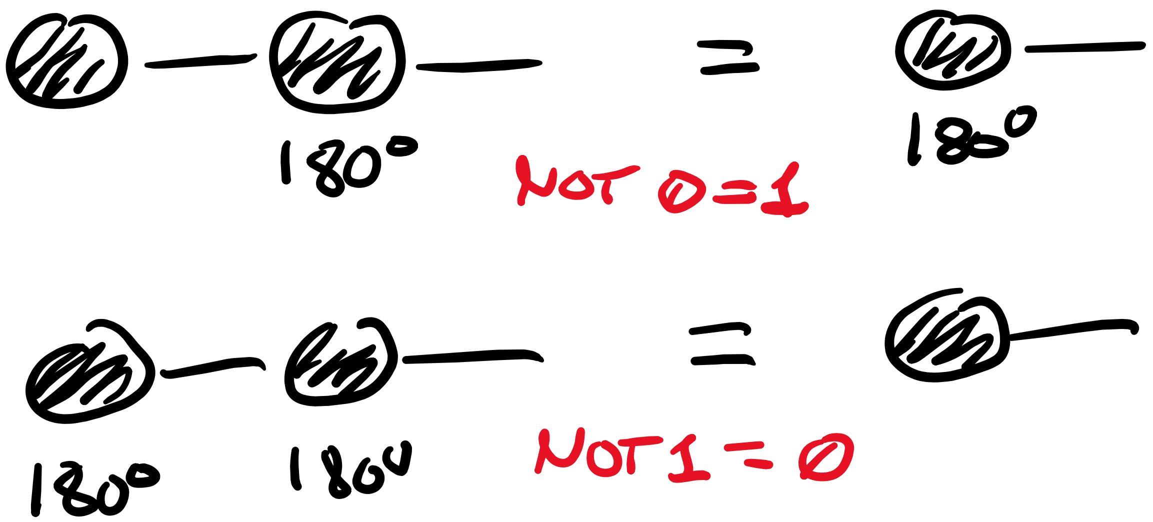
Spider arithmetic

Traditionally, the 0°/180° X spider states are the ones used to encode a bit, hence their name on the Bloch sphere:



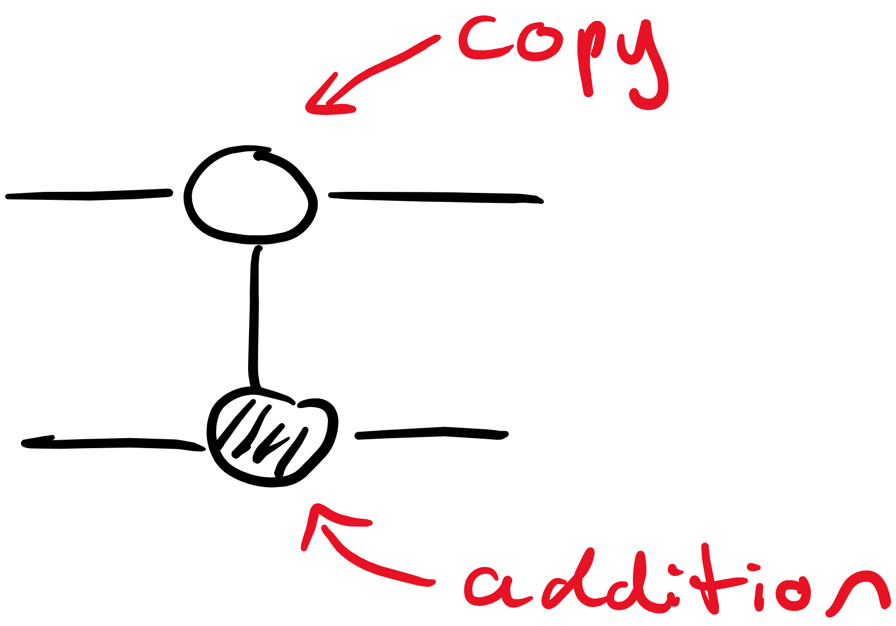
CNOTs from spiders

When using the 0°/180° X spider states to encode the bit values 0 and 1, the X gate acts as a NOT gate:

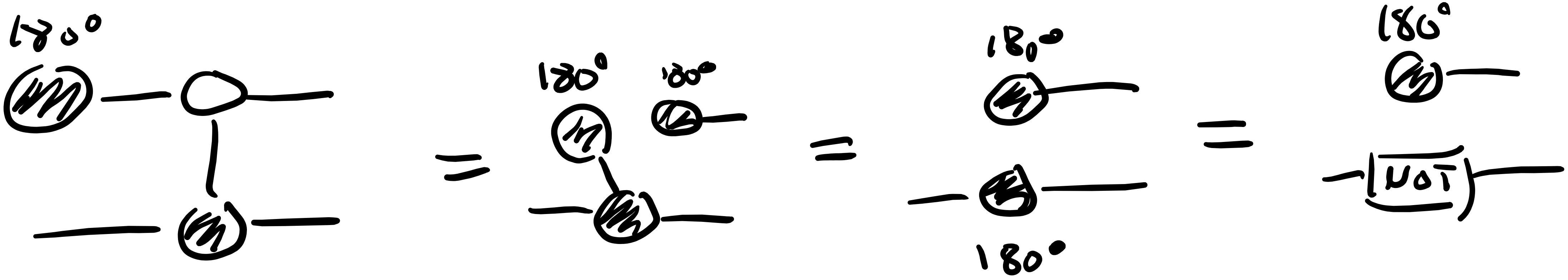


CNOTs from spiders

When using the 0°/180° X spider states to encode the bit values 0 and 1, the following acts as a CNOT gate:

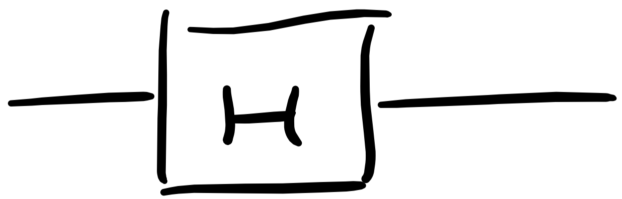


CNOTs from spiders



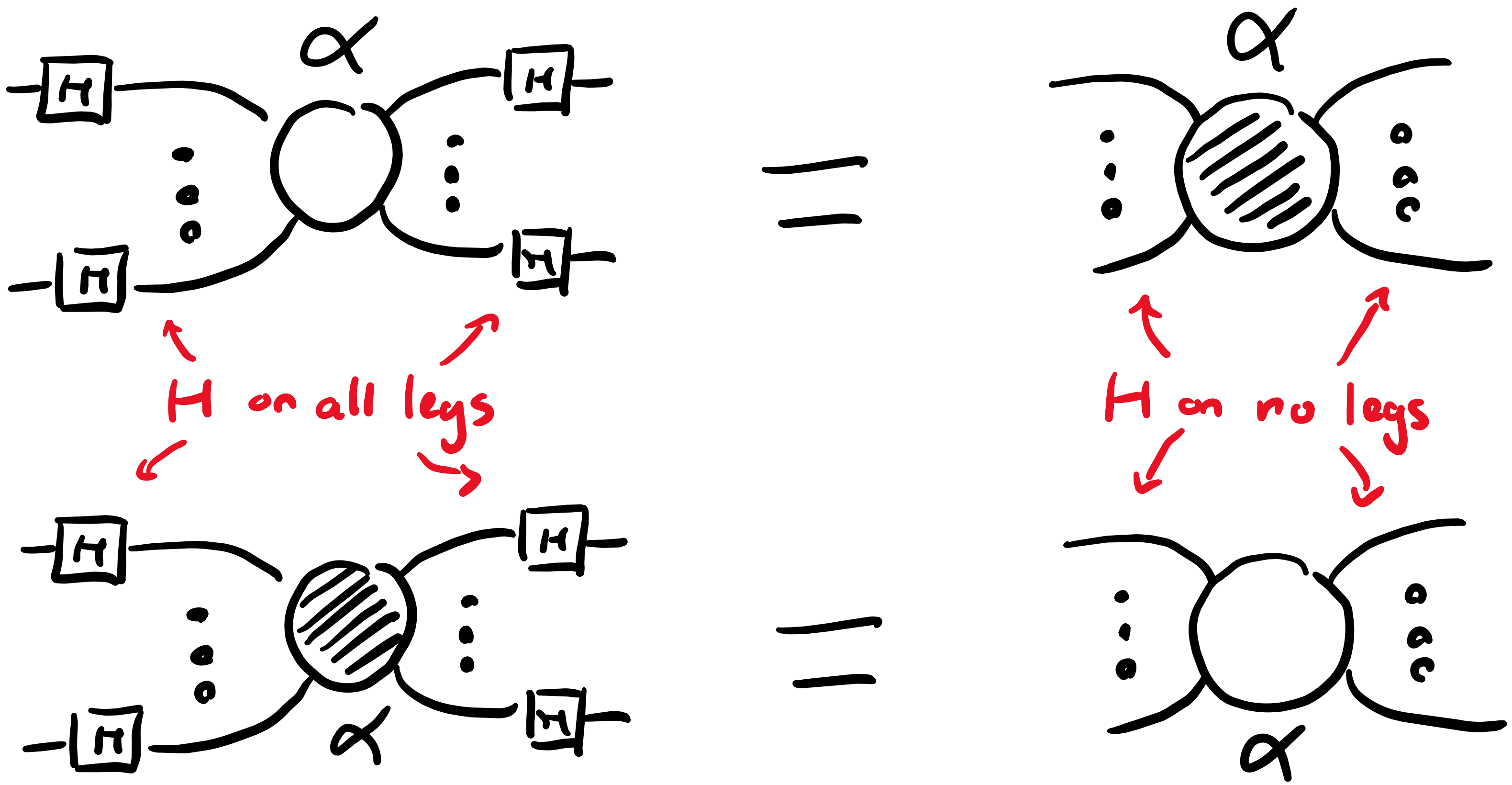
Colour-change gate

There is a special “colour-change” gate, also known as the “Hadamard gate”, which can be used to turn spiders of one colour into the other.



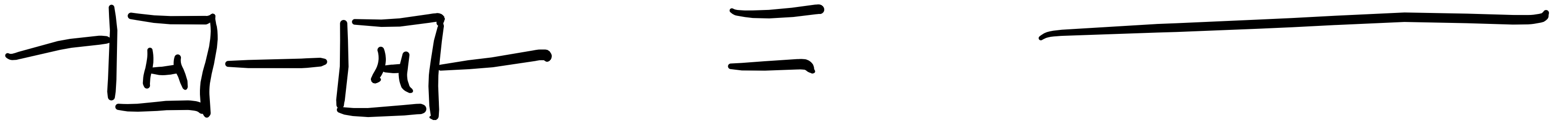
Hadamard gate

Colour-change rules



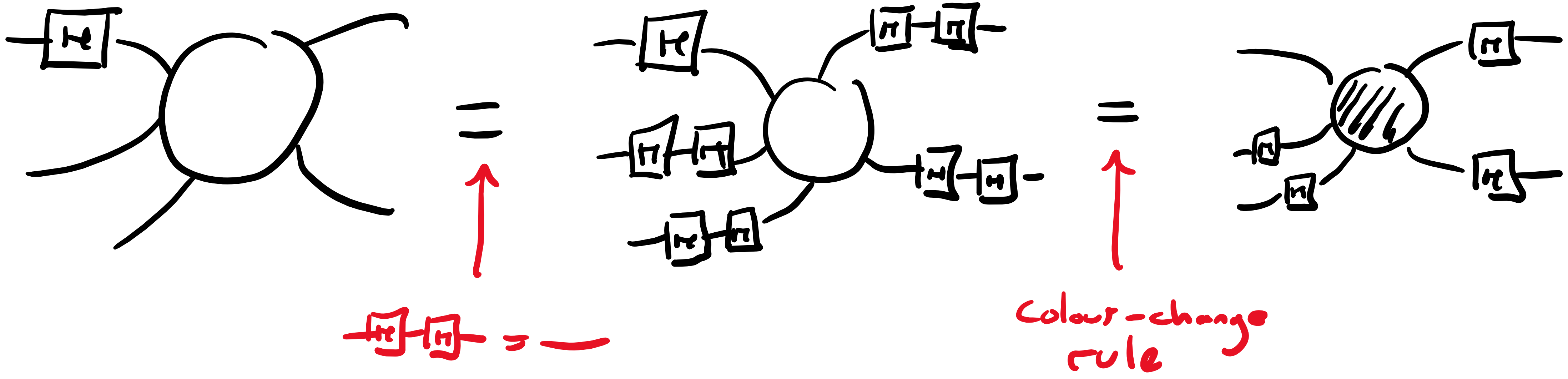
Colour-change gate

Applying the colour-change gate twice is like doing nothing at all.



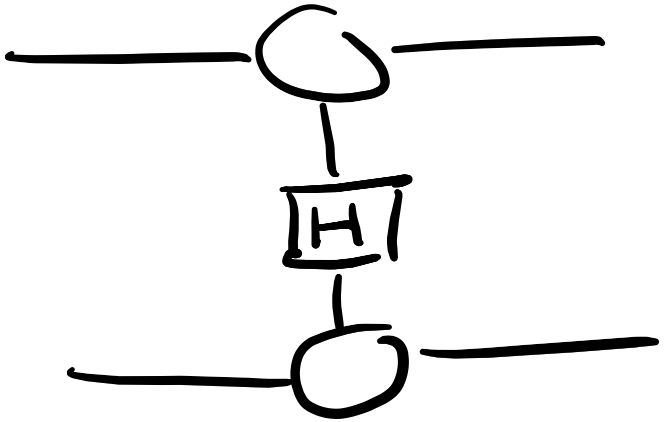
Derived colour-change rules

If we apply a colour change gate to one leg of a spider, it passes through, changes the spider colour and appears on all other legs:



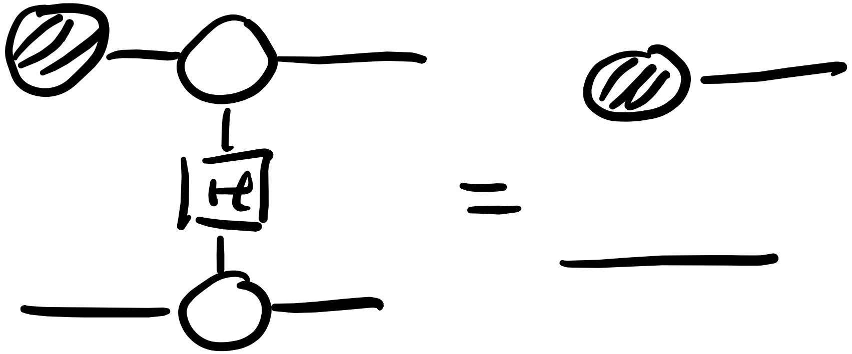
The CZ gate

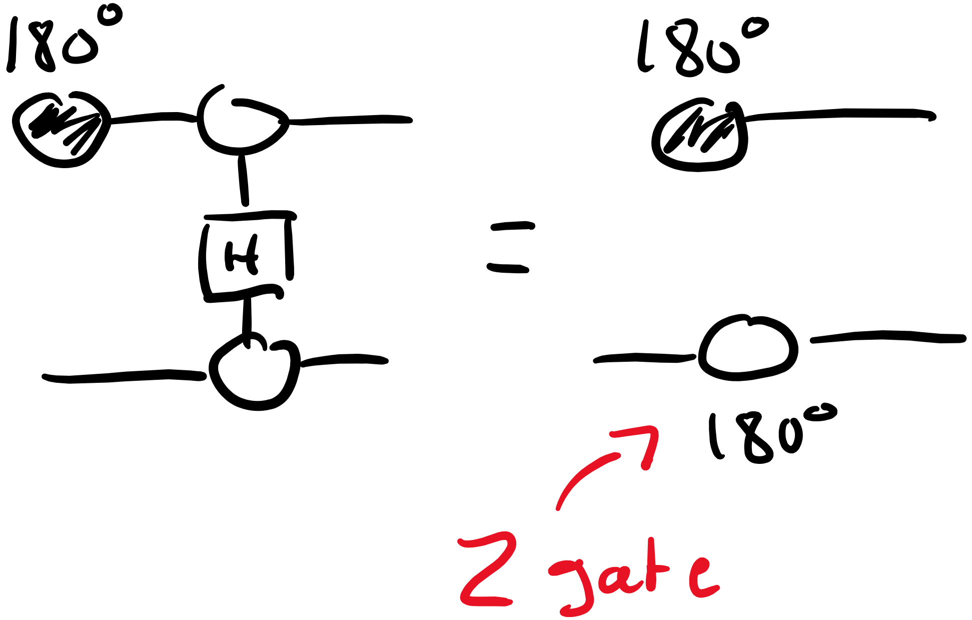
Because the X gate acts as NOT gate, the CNOT is sometimes called CX (for Controlled X). We can also make a CZ gate (for Controlled Z), where a Z gate is applied instead:



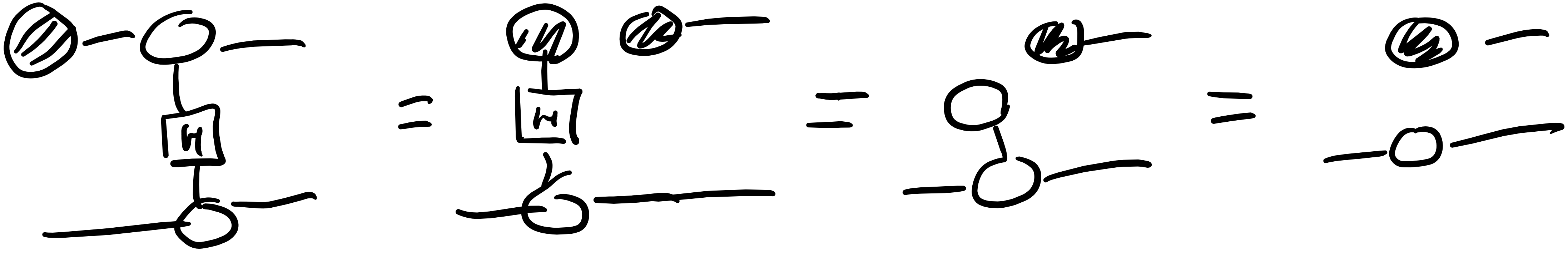
Exercise (5m)

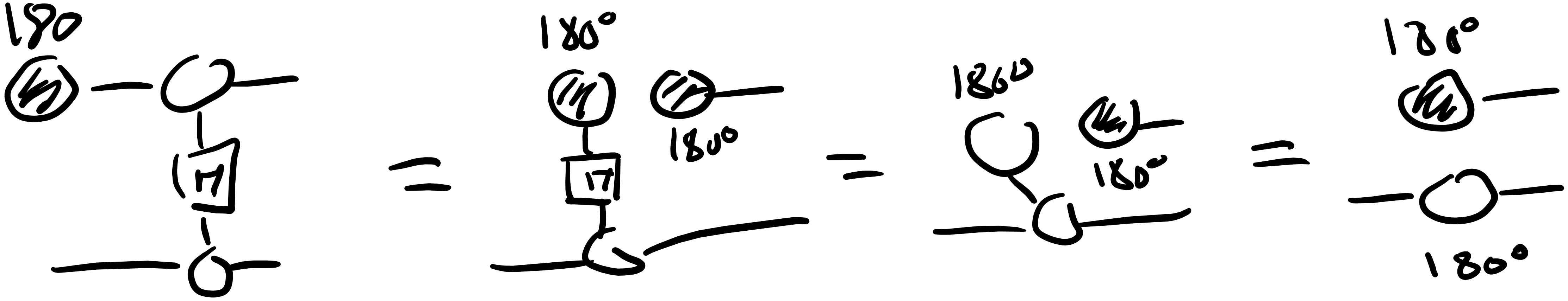
Use the copy rules, colour-change rule and spider fusion to prove that the gate defined in the previous slide is really the controlled Z gate:





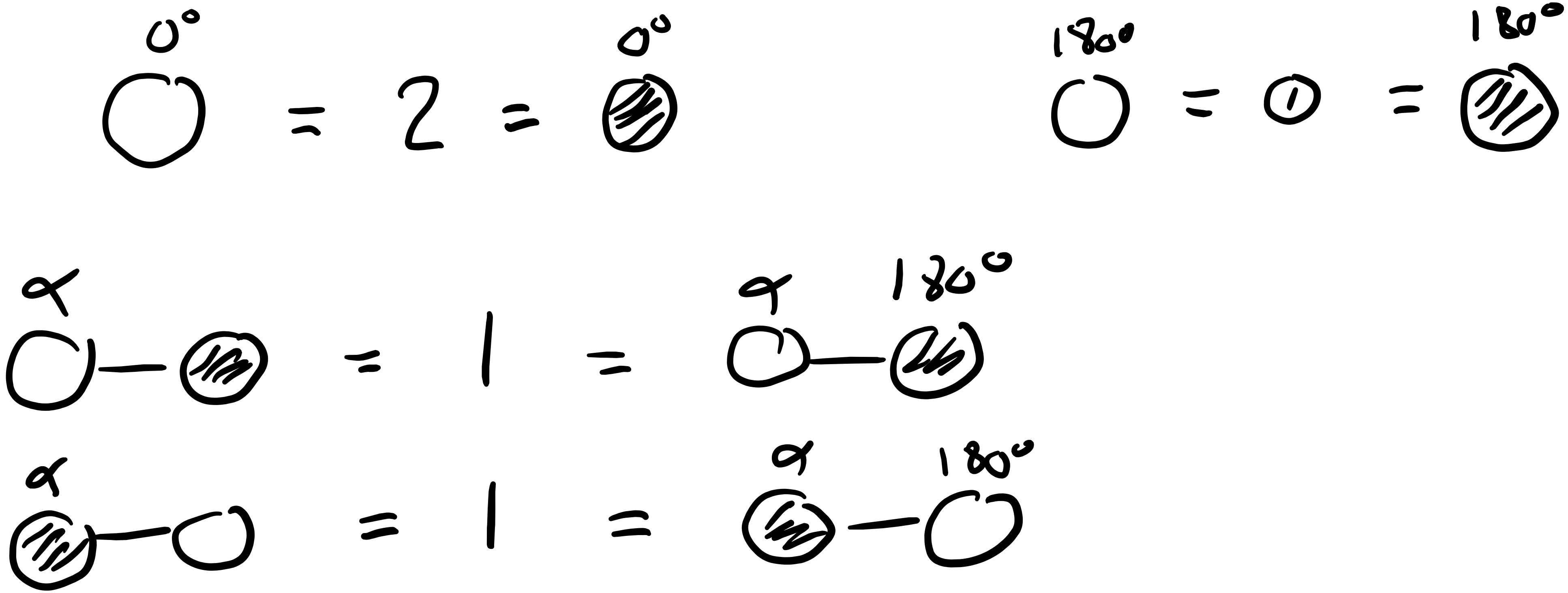
Exercise (Solutions)





Numbers

The final ingredient necessary to perform quantum computation are numbers, which are circuits with no inputs or outputs. Below are some simple numbers coming from spiders:



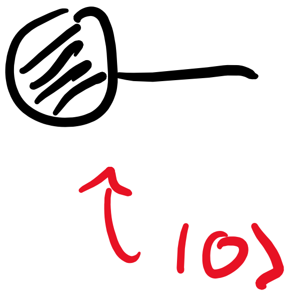
Measurement outcome probabilities

To compute the probabilities for a measurement outcome (a bitstring, 1 bit for each qubit), we do the following:

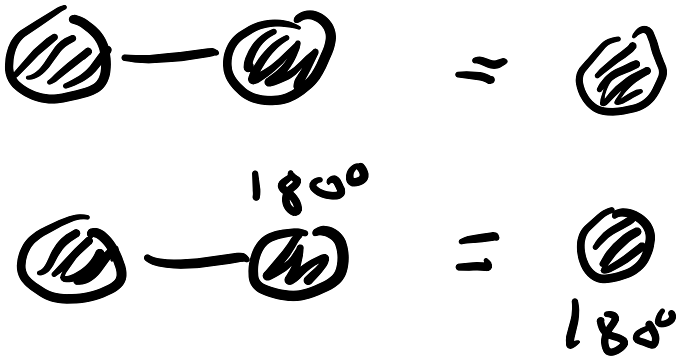
1. Apply a 0° or 180° X delete spider to each qubit (depending on whether the string is 0 or 1 at that qubit).
2. Simplify the circuit until we get to a number we know (let’s call it the “weight” for the bitstring).

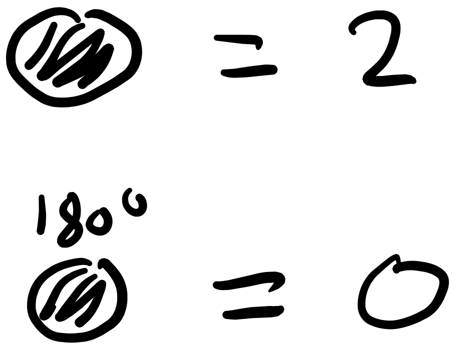
We then normalise all weights so that they sum to 1. Once normalised, those numbers are the probabilities we seek.

Measuring the $|0\rangle$ state



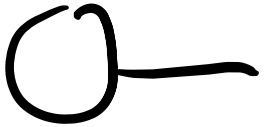
$$\Rightarrow \begin{cases} \text{outcome } 0 \\ \text{outcome } 1 \end{cases}$$





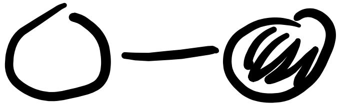
$$\Rightarrow \begin{cases} \mathbb{P}(0) = \dfrac{2}{2+0} = 1 \\[2ex] \mathbb{P}(1) = \dfrac{0}{2+0} = 0 \end{cases}$$

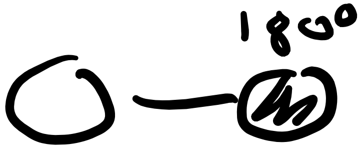
Measuring the $|+\rangle$ state

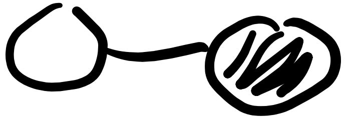


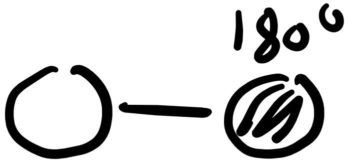
$|+\rangle$

$$\Rightarrow \begin{cases} \text{outcome } 0 \\ \text{outcome } 1 \end{cases}$$





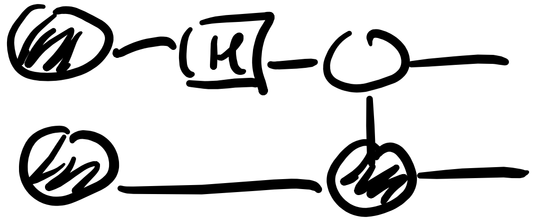
 $= 1$



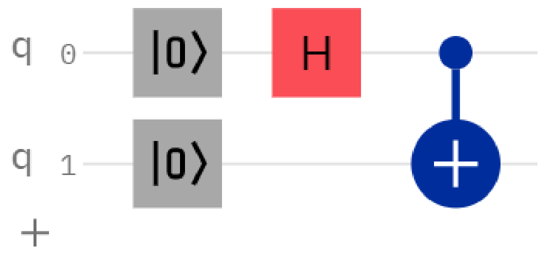
$= 1$

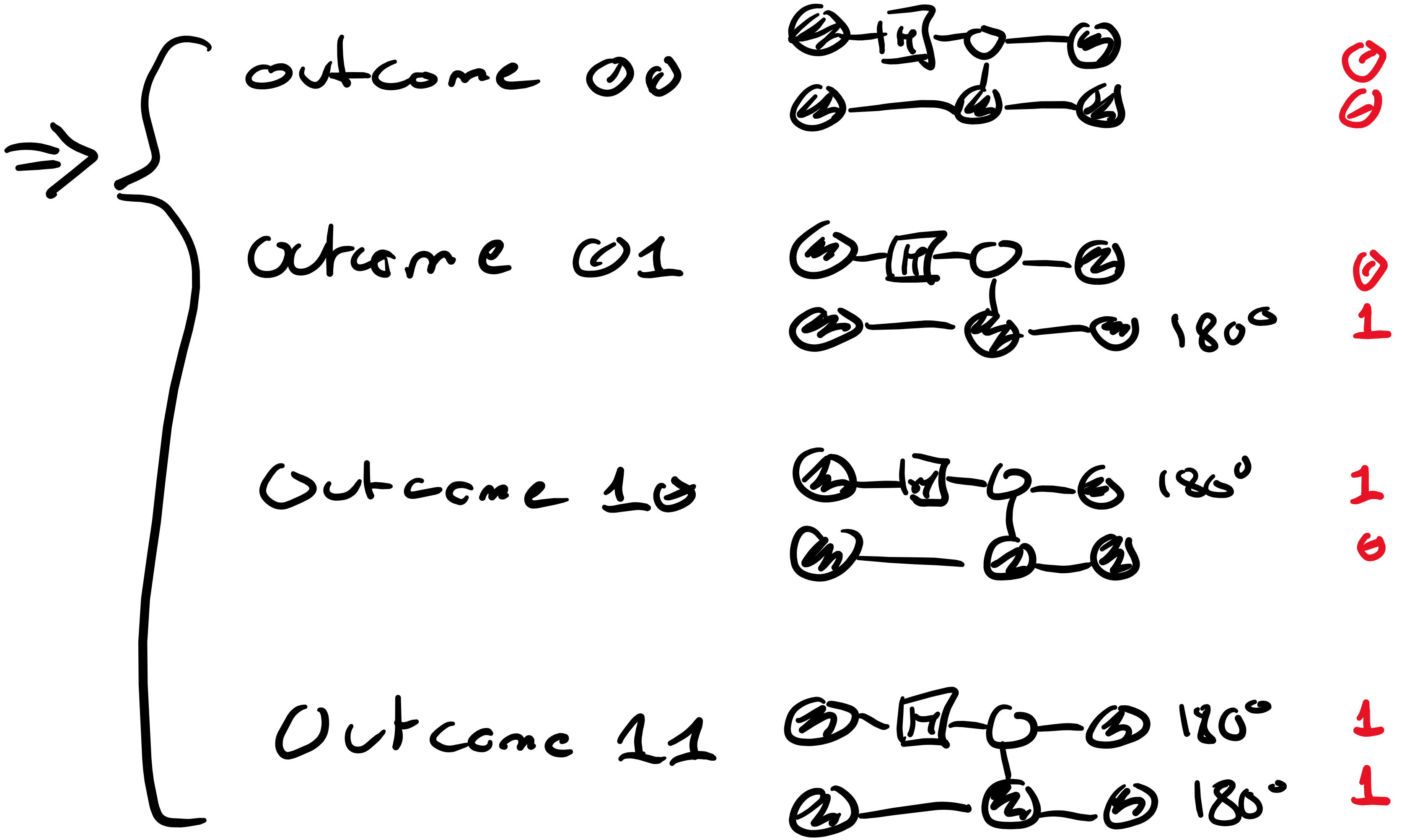
$$\Rightarrow \begin{cases} \mathbb{P}(0) = \frac{1}{1+1} = \frac{1}{2} \\ \mathbb{P}(1) = \frac{1}{1+1} = \frac{1}{2} \end{cases}$$

Measuring the Bell state

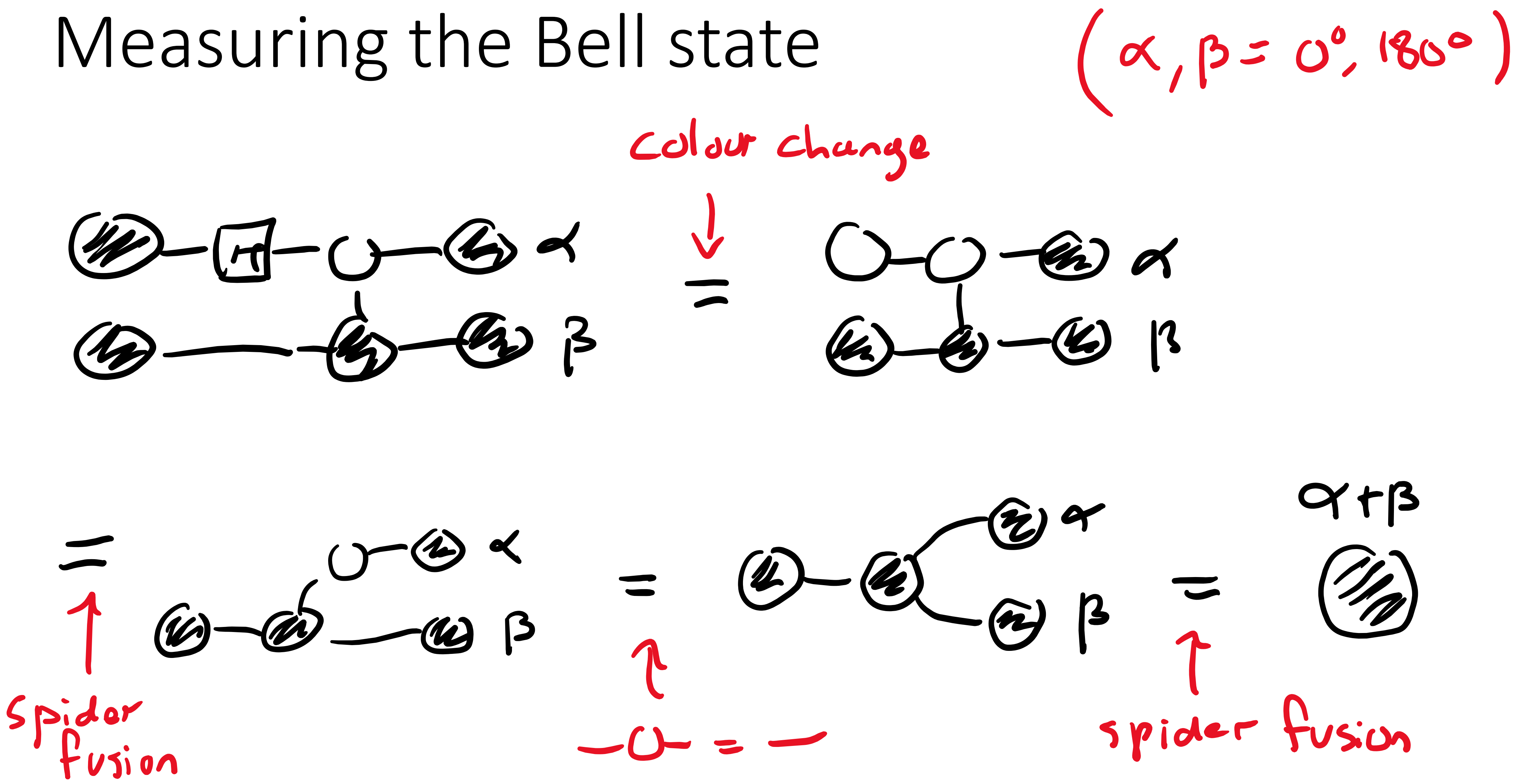


Circuit preparing the Bell state





Measuring the Bell state



Measuring the Bell state

Weight: $\alpha+\beta$

Outcome 00 : $\alpha=0^\circ, \beta=0^\circ \Rightarrow \alpha+\beta = \ = 2$

Outcome 01 : $\alpha=0^\circ, \beta=180^\circ \Rightarrow \alpha+\beta = 180^\circ = 0$

Outcome 10 : $\alpha=180^\circ, \beta=0^\circ \Rightarrow \alpha+\beta = 180^\circ = 0$

Outcome 11 : $\alpha=180^\circ, \beta=180^\circ \Rightarrow \alpha+\beta = \ = 2$

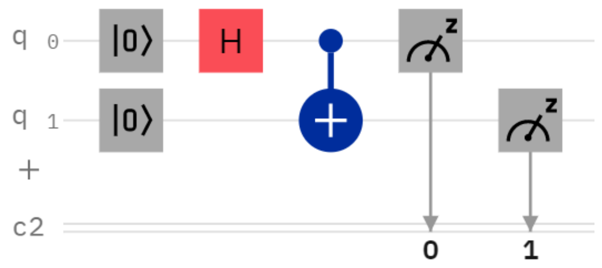
Measuring the Bell state

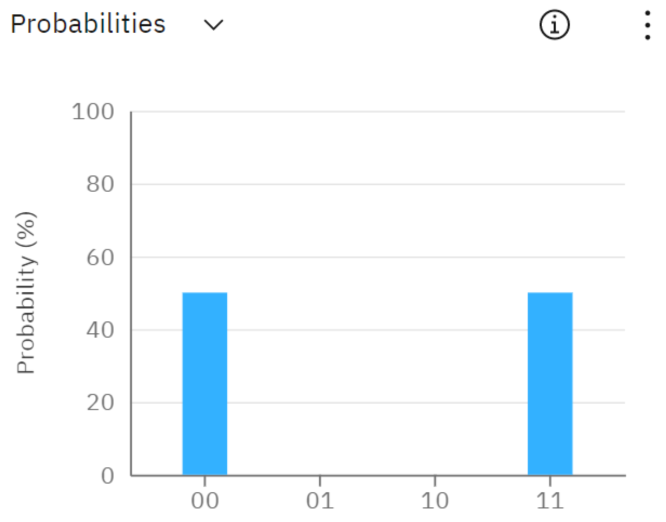
$$\mathbb{P}(00) = \frac{2}{2+0+0+2} = \frac{2}{4} = \frac{1}{2}$$

$$\mathbb{P}(01) = \frac{0}{2+0+0+2} = \frac{0}{4} = 0$$

$$\mathbb{P}(10) = \frac{0}{2+0+0+2} = \frac{0}{4} = 0$$

$$\mathbb{P}(11) = \frac{2}{2+0+0+2} = \frac{2}{4} = \frac{1}{2}$$





That's a wrap!

For any questions, please email
csrimasterclasses@cs.ox.ac.uk

Bob Coecke and Aleks Kissinger,
"Picturing Quantum Processes",
Cambridge University Press

Bob Coecke and Stefano Gogioso,
"Quantum Theory in Pictures",
Upcoming publication