Quantum teleportation, diagrams, and the one-time pad

A. Kissinger

Digital Security Group
Institute for Computing and Information Sciences
Radboud University Nijmegen

23rd November 2016
Outline

- Process theories
- Non-separability
- One-time pad
- Quantum teleportation
Outline

Process theories

Non-separability

One-time pad

Quantum teleportation
Processes

- A **process** is anything with zero or more *inputs* and zero or more *outputs*
Processes

- A **process** is anything with zero or more *inputs* and zero or more *outputs*
- For example, this **function**:

\[ f(x, y) = x^2 + y \]
Processes

- A **process** is anything with zero or more *inputs* and zero or more *outputs*
- For example, this function:

\[ f(x, y) = x^2 + y \]

...is a process when takes two real numbers as input, and produces a real number as output.
A process is anything with zero or more inputs and zero or more outputs.

For example, this function:

\[ f(x, y) = x^2 + y \]

...is a process when takes two real numbers as input, and produces a real number as output.

We could also write it like this:
Processes

• A **process** is anything with zero or more *inputs* and zero or more *outputs*

• For example, this **function**:

\[ f(x, y) = x^2 + y \]

...is a process when takes two real numbers as input, and produces a real number as output.

• We could also write it like this:
Processes

- A **process** is anything with zero or more *inputs* and zero or more *outputs*
- For example, this function:
  \[ f(x, y) = x^2 + y \]
  ...is a process when takes two real numbers as input, and produces a real number as output.
- We could also write it like this:

\[
\begin{array}{c}
\text{R} \\
\hline
f \\
\hline
\text{R} \quad \text{R} \\
\end{array}
\]

- The labels on wires are called **system-types** or just **types**
More processes

• Similarly, a computer programs are processes
More processes

- Similarly, a computer program is a process.
- For example, a program that sorts lists might look like this:

```
<table>
<thead>
<tr>
<th>lists</th>
</tr>
</thead>
<tbody>
<tr>
<td>quicksort</td>
</tr>
<tr>
<td>lists</td>
</tr>
</tbody>
</table>
```
More processes

- Similarly, a computer program is a process.
- For example, a program that sorts lists might look like this:

```
<table>
<thead>
<tr>
<th>lists</th>
</tr>
</thead>
<tbody>
<tr>
<td>quicksort</td>
</tr>
<tr>
<td>lists</td>
</tr>
</tbody>
</table>
```

- These are also perfectly good processes:

```
<table>
<thead>
<tr>
<th>light</th>
<th>light</th>
</tr>
</thead>
<tbody>
<tr>
<td>binoculars</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>light</th>
<th>light</th>
</tr>
</thead>
<tbody>
<tr>
<td>cooking</td>
<td></td>
</tr>
</tbody>
</table>

| breakfast |
| light | light |
| eggs | bacon |

<table>
<thead>
<tr>
<th>noise</th>
<th>poo</th>
</tr>
</thead>
<tbody>
<tr>
<td>baby</td>
<td></td>
</tr>
</tbody>
</table>

| food | love |
Diagrams

• We can combine simple processes to make more complicated ones, described by diagrams:

\[
\begin{array}{c}
A & A & C \\
\downarrow & \downarrow & \downarrow \\
g & D \\
\downarrow & \downarrow \\
f & h \\
\end{array}
\]

\[k = f \cdot g = h \cdot g\]
• We can combine simple processes to make more complicated ones, described by diagrams:

\[
\begin{align*}
A & \xrightarrow{g} C \\
A & \xrightarrow{B} g \\
B & \xrightarrow{D} h \\
C & \xrightarrow{A} h
\end{align*}
\]

• The golden rule: only connectivity matters!

\[
\begin{align*}
\text{f} & \xrightarrow{g} \text{h} \\
\text{g} & \xrightarrow{k} \text{h} \\
\text{h} & \xrightarrow{f} \text{g} \\
\text{f} & \xrightarrow{g} \text{f} \\
\text{g} & \xrightarrow{k} & \text{h} \\
\text{h} & \xrightarrow{k} & \text{h}
\end{align*}
\]
• Special cases are parallel composition:
Diagrams

...and **sequential composition**:

\[
\begin{pmatrix}
E & F \\
g & f \\
C & D
\end{pmatrix}
\circ
\begin{pmatrix}
C & D \\
a & b \\
A & B
\end{pmatrix} =
\begin{pmatrix}
E & F \\
g & f \\
a & b \\
A & B
\end{pmatrix}
\]
Types

- Connections are only allowed where the types match
• Connections are only allowed where the types match, e.g.:
Connections are only allowed where the types match, e.g.:
• Connections are only allowed where the *types match*, e.g.:

1. Valid: $A - h - C, D - g - D$
2. Valid: $D - h - A, C - g - D$
3. Invalid: $A - h - A \neq C, D - g - D$
Types and Process Theories

- Types tell us when it makes sense to plug processes together
Types and Process Theories

- Types tell us when it makes sense to plug processes together
- Ill-typed diagrams are undefined:

```
noise

?  

quicksort


poo


baby


food


love
```
Types and Process Theories

- Types tell us when it makes sense to plug processes together.
- Ill-typed diagrams are undefined:

```
noise ?

quicksort

baby

food love

poo
```

- In fact, these processes don’t ever sense to plug together.
Types and Process Theories

- Types tell us when it makes sense to plug processes together.
- Ill-typed diagrams are undefined:

![Diagram showing connections between 'noise', 'quicksort', 'poo', 'baby', 'food', and 'love'.]

- In fact, these processes don’t ever sense to plug together.
- A family of processes which do make sense together is called a process theory.
Example: relations

In the process theory of relations:
Example: relations

In the process theory of relations:

- system-types are sets
Example: relations

In the process theory of relations:

- system-types are sets
- processes are relations

\[
\begin{align*}
\{x, y, z\} &\quad R &\quad \{a, b, c\} \\
\begin{array}{c}
R \\
\{a, b, c\}
\end{array} &\quad = &\quad \left\{ \begin{array}{c}
a \mapsto x \\
a \mapsto y \\
b \mapsto z
\end{array} \right.
\end{align*}
\]
Example: relations

In the process theory of relations:

- system-types are sets
- processes are relations

\[
\begin{align*}
R & \colon \{x, y, z\} \rightarrow \{a, b, c\} \\
\{x, y, z\} & \rightarrow \{a \rightarrow x, a \rightarrow y, b \rightarrow z\} = \\
\{a, b, c\} & \rightarrow \emptyset
\end{align*}
\]

...which we can think of as non-deterministic computations:
Example: relations

Relations compose in *sequentially* just like you learned in school:

\[ R \circ S \]
Example: relations

...and they compose in parallel via the cartesian product.
Example: relations

...and they compose in parallel via the cartesian product.

- that is, systems compose like this:

\[
\begin{array}{c|c}
A & B \\
\end{array}
\]
Example: relations

...and they compose in parallel via the cartesian product.

- that is, systems compose like this:

\[
\begin{array}{c|c}
A & B \\
\hline
\end{array}
\begin{array}{l}
:= \{(a, b) \mid a \in A, b \in B\}
\end{array}
\]
Example: relations

...and they compose in parallel via the cartesian product.

- that is, systems compose like this:

\[
\begin{align*}
A \times B &= \{(a, b) \mid a \in A, b \in B\}
\end{align*}
\]

- so relations compose like this:

\[
\begin{align*}
R \times S \colon (a, b) &\mapsto (c, d) \iff \\
R \colon a &\mapsto c \text{ and } S \colon b &\mapsto d
\end{align*}
\]
Some processes in relations

- ‘no wire’ is a one-element set:

\[
\text{:= } \{\bullet\}
\]
Some processes in relations

- ‘no wire’ is a one-element set:

\[ \{\bullet\} \]

- ...because:

\[
\begin{array}{c}
A \quad = \\
\{(a, \bullet) \mid a \in A\} \cong A = A
\end{array}
\]
Some processes in relations

- ‘no wire’ is a one-element set:

\[
\begin{array}{c}
\begin{array}{c}
\text{•} \\
\end{array}
\end{array} := \{\bullet\}
\]

- ...because:

\[
A = \{(a, \bullet) \mid a \in A\} \cong A = A
\]

- processes from ‘no wire’ represent (non-deterministic) states
Some processes in relations

- ‘no wire’ is a one-element set:
  \[
  \square := \{ \bullet \}
  \]

- ...because:
  \[
  A \quad = \quad \{(a, \bullet) \mid a \in A\} \cong A = A
  \]

- processes from ‘no wire’ represent (non-deterministic) states, e.g. for a bit:
  \[
  \begin{array}{c}
  \downarrow 0 \\
  \bullet 
  \end{array} \quad = \quad \{ \bullet \mapsto 0 \}
Some processes in **relations**

- ‘no wire’ is a one-element set:

  \[
  \begin{array}{c}
  \text{no wire} \\
  \hline
  \end{array}
  \quad := \quad \{\bullet\}
  \]

- ...because:

  \[
  A = \{(a, \bullet) \mid a \in A\} \cong A = A
  \]

- processes **from** ‘no wire’ represent (non-deterministic) states, e.g. for a bit:

  \[
  \begin{align*}
  \begin{array}{c}
  \text{0} \\
  \hline
  \end{array} & = \{\bullet \mapsto 0\} \\
  \begin{array}{c}
  \text{1} \\
  \hline
  \end{array} & = \{\bullet \mapsto 1\}
  \end{align*}
  \]
Some processes in relations

- 'no wire' is a one-element set:
  \[ : = \{\bullet\}\]

- ...because:
  \[ A = \{(a, \bullet) \mid a \in A\} \cong A = A \]

- processes from 'no wire' represent (non-deterministic) states, e.g. for a bit:
  \[ 0 = \{\bullet \mapsto 0\} \quad 1 = \{\bullet \mapsto 1\} \quad * = \{\bullet \mapsto \{0, 1\}\} \]
Some processes in relations

- ...whereas processes to ‘no wire’ are called effects.
Some processes in relations

- ...whereas processes to ‘no wire’ are called effects. These test for the given state(s):

\[ \triangle = \left\{ 0 \leftrightarrow \bullet \right\} \]
Some processes in **relations**

- ...whereas processes to ‘no wire’ are called **effects**. These test for the given state(s):

\[
\begin{align*}
\begin{array}{c}
\triangleleft 0 \\
\downarrow 0 \\
\end{array} & = \{0 \leftrightarrow \bullet\} \\
\begin{array}{c}
\triangleleft 1 \\
\downarrow 1 \\
\end{array} & = \{1 \leftrightarrow \bullet\}
\end{align*}
\]
Some processes in relations

• ...whereas processes to ‘no wire’ are called effects. These test for the given state(s):

\[
\begin{align*}
\blacktriangleleft 0 & = \{ 0 \leftrightarrow \bullet \} \\
\blacktriangleleft 1 & = \{ 1 \leftrightarrow \bullet \} \\
\blacktriangleleft \ast & = \{ \{0, 1\} \leftrightarrow \bullet \}
\end{align*}
\]

• when state meets effect, there are two possibilities:

\[
\begin{align*}
\blacktriangledown T \blacktriangledown S & = \{ \bullet \leftrightarrow \bullet \} \\
\blacktriangledown T \blacktriangledown S & = \emptyset
\end{align*}
\]
Some processes in relations

- ...whereas processes to ‘no wire’ are called effects. These test for the given state(s):

\[
\begin{align*}
\triangleleft 0 &= \{0 \leftrightarrow \bullet\} \\
\triangleleft 1 &= \{1 \leftrightarrow \bullet\} \\
\triangleleft \ast &= \{\{0, 1\} \leftrightarrow \bullet\}
\end{align*}
\]

- when state meets effect, there are two possibilities:

\[
\begin{align*}
\triangleleft T \neg \neg S &= \{\bullet \leftrightarrow \bullet\} \\
\triangleleft T \neg \neg S &= \emptyset
\end{align*}
\]

These stand for true and false.
States on two systems

- States on two systems are more interesting
States on two systems

- States on two systems are more interesting, e.g.:

\[
\psi := \{ * \mapsto \{(0,0), (1,1)\} \}
\]
States on two systems

- States on two systems are more interesting, e.g.:

\[ \psi := \left\{ \ast \mapsto \{ (0, 0), (1, 1) \} \right\} \]

**Interpretation:** “I don’t know what bit I have, but I know its the same as yours”
States on two systems

- States on two systems are more interesting, e.g.:

\[
\psi := \{ \ast \mapsto (0,0), (1,1) \}
\]

**Interpretation:** “I don’t know what bit I have, but I know it’s the same as yours”

- States of the two systems no longer have their own, separate identities
States on two systems

- States on two systems are more interesting, e.g.:

\[
\psi := \{(0, 0), (1, 1)\}
\]

**Interpretation:** “I don’t know what bit I have, but I know its the same as yours”

- States of the two systems no longer have their own, separate identities

- Hence we get...
Outline

Process theories

Non-separability

One-time pad

Quantum teleportation
Separable states

- A state $\psi$ on two systems is separable if there exist $\psi_1, \psi_2$ such that:

\[
\psi = \psi_1 \psi_2
\]
A state $\psi$ on two systems is *separable* if there exist $\psi_1$, $\psi_2$ such that:

$$\psi = \psi_1 \psi_2$$

**Intuitively:** the properties of the system on the left are *independent* from those on the right
A state $\psi$ on two systems is **separable** if there exist $\psi_1, \psi_2$ such that:

\[
\psi = \psi_1 \otimes \psi_2
\]

**Intuitively:** the properties of the system on the left are *independent* from those on the right.

In the deterministic-land, *all states* to separate...
Characterising non-separability

• ...which is why non-separable states are way more interesting!
Characterising non-separability

• ...which is why non-separable states are way more interesting!
• But, how do we know we’ve found one?
...which is why non-separable states are way more interesting!

But, how do we know we’ve found one?

i.e. that there do not exist states $\psi_1, \psi_2$ such that:

\[
\begin{array}{c}
\psi \\
\end{array} = \begin{array}{c}
\psi_1 \\
\end{array} \begin{array}{c}
\psi_2 \\
\end{array}
\]
• ...which is why non-separable states are way more interesting!
• But, how do we know we’ve found one?
• i.e. that there do not exist states $\psi_1, \psi_2$ such that:

$$\psi = \psi_1 \psi_2$$

• **Problem:** Showing that something *doesn’t* exist is hard.
Solution: Replace a negative property with a positive one:
Solution: Replace a negative property with a positive one:

Definition
A state $\psi$ is called \textit{cup-state} if there exists an effect $\phi$, called a \textit{cap-effect}, such that:

\[
\phi \psi = \psi \phi
\]
Cup-states

- By introducing some clever notation:

\[
\begin{align*}
\cup \quad &:= \quad \begin{tikzpicture}
    \draw (0,0) -- (0,1);
    \draw (0,0) -- (0,2);
    \draw (0,0) -- (0,3);
    \node at (0,2) {$\psi$};
\end{tikzpicture} \\
\cap \quad &:= \quad \begin{tikzpicture}
    \draw (0,0) -- (0,1);
    \draw (0,1) -- (0,2);
    \draw (0,2) -- (0,3);
    \node at (0,1) {$\phi$};
\end{tikzpicture}
\end{align*}
\]
Cup-states

- By introducing some clever notation:

\[
\begin{align*}
\bigcirc & := \begin{equation}
\psi
\end{equation} \\
\bigcap & := \begin{equation}
\phi
\end{equation}
\end{align*}
\]

- Then these equations:

\[
\begin{align*}
\begin{equation}
\phi
\end{equation} & = \begin{equation}
\psi
\end{equation} \\
\begin{equation}
\phi
\end{equation} & = \begin{equation}
\psi
\end{equation}
\end{align*}
\]
Cup-states

- By introducing some clever notation:

\[ \begin{array}{cccc}
\cup & := & \psi \\
\psi & := & \phi \\
\end{array} \]

- Then these equations:

\[ \begin{array}{cccc}
\phi & \psi & = & = \\
\psi & \phi & = & \psi \\
\end{array} \]

- ...look like this:
Yank the wire!
Yank the wire!
Example

- In **relations**, there is an obvious choice of cup-state:

\[
\cup := \{ \ast \mapsto \{(0,0), (1,1)\} \}
\]
Example

• In relations, there is an obvious choice of cup-state:

\[
\bigcirc := \{ \ast \mapsto \{(0, 0), (1, 1)\} \}
\]

• The associated cap-effect corresponds to “checking if two bits are the same”:

\[
\bigotimes := \{ \{(0, 0), (1, 1)\} \mapsto \ast \}
\]
Example

- In **relations**, there is an obvious choice of cup-state:

\[
\bigtriangleup := \{ * \mapsto \{(0, 0), (1, 1)\}\}
\]

- The associated cap-effect corresponds to “checking if two bits are the same”:

\[
\bigtriangledown := \{\{(0, 0), (1, 1)\}\ \mapsto \ast\}
\]

- This, plus NOT...  

\[
\text{NOT} := \left\{ \begin{array}{c}
0 \mapsto 1 \\
1 \mapsto 0
\end{array} \right.
\]

...gives us enough to start building interesting stuff.
Outline

Process theories
Non-separability
One-time pad
Quantum teleportation
An incredibly sophisticated security protocol

- Suppose Aleks and Bob each have an envelope with the same (random) bit sealed inside
An incredibly sophisticated security protocol

- Suppose Aleks and Bob each have an envelope with the same (random) bit sealed inside
- Aleks wants to send a bit to Bob, but is paranoid (as usual)
An incredibly sophisticated security protocol

- Suppose Aleks and Bob each have an envelope with the same (random) bit sealed inside.
- Aleks wants to send a bit to Bob, but is paranoid (as usual).
- He opens his envelope, and tells Bob if the bit inside is the same as the one he wants to send.
An incredibly sophisticated security protocol

• Suppose Aleks and Bob each have an envelope with the same (random) bit sealed inside
• Aleks wants to send a bit to Bob, but is paranoid (as usual)
• He opens his envelope, and tells Bob if the bit inside is the same as the one he wants to send
• Bob opens his envelope, and:
• Suppose Aleks and Bob each have an envelope with the same (random) bit sealed inside
• Aleks wants to send a bit to Bob, but is paranoid (as usual)
• He opens his envelope, and tells Bob if the bit inside is the same as the one he wants to send
• Bob opens his envelope, and:
  • if the bits matched before, Bob now has Aleks’ bit,
An incredibly sophisticated security protocol

- Suppose Aleks and Bob each have an envelope with the same (random) bit sealed inside
- Aleks wants to send a bit to Bob, but is paranoid (as usual)
- He opens his envelope, and tells Bob if the bit inside is the same as the one he wants to send
- Bob opens his envelope, and:
  - if the bits matched before, Bob now has Aleks’ bit,
  - otherwise he flips the bit.
One-time pad with relations

- we can represent the envelopes with the shared random bit as a cup-state:

\[
\bigcup := \{ \ast \mapsto \{(0, 0), (1, 1)\} \}
\]
One-time pad with relations

- we can represent the envelopes with the shared random bit as a cup-state:

\[
\bigcirc := \{ * \mapsto (0, 0), (1, 1) \}
\]

- then checking whether two bits are the same is a ‘measurement’ that Aleks can perform on his systems
One-time pad with relations

• we can represent the envelopes with the shared random bit as a cup-state:

\[
\cup := \{ * \mapsto \{(0, 0), (1, 1)\}\}
\]

• then checking whether two bits are the same is a ‘measurement’ that Aleks can perform on his systems

• There are two possible outcomes:

\[
\begin{align*}
\cup &:= \text{“the same”} , \\
\text{NOT} &:= \text{“NOT the same”}
\end{align*}
\]
One-time pad with relations

- ...which we can write as:

\[
\begin{cases}
  U_i \\
\end{cases} \quad i \in \{0,1\}
\]

\[
\begin{align*}
U_0 & := \text{NOT} \\
U_1 & := \text{NOT}
\end{align*}
\]
One-time pad with relations

- ...which we can write as:

\[
\begin{cases}
U_i \\
\end{cases}
\quad i \in \{0, 1\}
\]

- Then, the \( U_i \) satisfy:

\[
U_i = \text{NOT}
\]
So, the OTP protocol looks like this:
So, the OTP protocol looks like this:

```
P = \psi

U_i

```

Aleks' "measurement" → Bob's fix

Bob's fix → envelope 1

envelope 1 → envelope 2

envelope 2 → Aleks' bit

Aleks' bit → \psi

\psi → U_i

U_i → Bob
...and it works

![Diagram showing quantum teleportation process with labels for Aleks and Bob, and gates $U_i$ and $b$.]
...and it works
...and it works

\[ \begin{align*}
&\text{Aleks} \\
&U_i \\
&U_i \\
&b \\
&\text{Bob}
\end{align*} \]
...and it works

- Aleks
- Bob

Quantum teleportation, diagrams, and the one-time pad
Outline

Process theories

Non-separability

One-time pad

Quantum teleportation
Quantum bits

- We go from classical to quantum by changing the process theory:
  
  \[ \text{relations} \Rightarrow \text{quantum maps} \]
Quantum bits

- We go from classical to quantum by changing the process theory:
  \[ \text{relations} \Rightarrow \text{quantum maps} \]
- The quantum analogue to a bit is a \textit{qubit}, which represents the state of the simplest non-trivial quantum system.
Quantum bits

- We go from classical to quantum by changing the process theory:

  \[ \text{relations} \Rightarrow \text{quantum maps} \]

- The quantum analogue to a bit is a qubit, which represents the state of the simplest non-trivial quantum system
- Example: polarization of a photon
Quantum bits

- The state space of a bit consists of two points: 0 and 1
Quantum bits

- The state space of a bit consists of two points: 0 and 1
- ...whereas qubits, it forms a sphere:
Quantum bits

- The state space of a bit consists of two points: 0 and 1
- ...whereas qubits, it forms a sphere:

```
|ψ⟩
\alpha
\theta
1
```

- “Plain old” bits live at the North Pole and the South Pole.
- In quantum-land, we can realise a ‘cup’ using quantum entanglement

\[ \text{\textendash\textendash\textendash} \text{“Bell state”} \]
Quantum entanglement

- In quantum-land, we can realise a ‘cup’ using quantum entanglement

  \( \bigcirc \leftrightarrow \text{“Bell state”} \)

- Even though this thing is (slightly) more complicated to describe, it acts just like before
Quantum measurement

- We also have a quantum analogue for Aleks’ measurement:

\[
\begin{array}{c}
\{U_i\} \\
i \in \{0,1,2,3\} \\
\end{array}
\]

\[\iff \text{“Bell measurement”}\]
• We also have a quantum analogue for Aleks’ measurement:

\[
\begin{cases}
U_i \\
\end{cases}
\quad i \in \{0, 1, 2, 3\}
\]

⇐ “Bell measurement”

where there are now three different ways to “NOT”:

\[
\begin{align*}
U_0 & := \quad \\
U_2 & := \quad \\
U_1 & := \quad \\
U_3 & := \quad 
\end{align*}
\]
OTP $\Rightarrow$ quantum teleportation

**Diagram**: Diagram showing the process of quantum teleportation. The diagram includes:
- A label for Aleks' "measurement" pointing to a symbol $U_i$.
- An arrow leading from the symbol $U_i$ to a symbol $\psi$ labeled Aleks' bit.
- A looped arrow from $\psi$ to $U_i$.
- Arrows labeled "envelope 1" and "envelope 2" leading to Bob's fix.
- A label for Bob's fix pointing back to $U_i$.

**Text**: 
- Process theories
- Non-separability
- One-time pad
- Quantum teleportation
- Radboud University Nijmegen

**Note**: The diagram illustrates the quantum teleportation process from a one-time pad (OTP) scenario, highlighting the key steps in the quantum teleportation protocol.
OTP $\Rightarrow$ quantum teleportation

Bell measurement $\Rightarrow$ quantum state $\Rightarrow$ Bell state

Bob's fix

A. Kissinger
23rd November 2016
Quantum teleportation, diagrams, and the one-time pad
...and it works
...and it works

\[ \psi \]

\[ U_i \]

Aleks

Bob

\[ U_i \]
...and it works

\[ \psi \]

\[ U_i \]

\[ U_i \]

Aleks

Bob
...and it works

A. Kissinger
23rd November 2016
Two for the price of one

- **The moral:** In both OTP and teleportation, Aleks must send Bob $i$, otherwise the whole thing fails.
Two for the price of one

- **The moral**: In both OTP and teleportation, Aleks must send Bob $i$, otherwise the whole thing fails.

- By using a *shared resource*:

  $\bigcirc := \text{shared random bit}$

  $\bigotimes := \text{Bell state}$
Two for the price of one

- **The moral**: In both OTP and teleportation, Aleks must send Bob $i$, otherwise the whole thing fails.
- By using a **shared resource**:
  
  $\bigcirc := \text{shared random bit}$  
  
  $\bigotimes := \text{Bell state}$

- Aleks can send **one kind of thing**:

  $i \in \{0, 1\} := \text{public data}$  
  
  $i \in \{0, 1, 2, 3\} := \text{classical data}$
Two for the price of one

- **The moral:** In both OTP and teleportation, Aleks must send Bob $i$, otherwise the whole thing fails.
- By using a **shared resource**:
  $$\bigotimes := \text{shared random bit}$$
  $$\bigotimes := \text{Bell state}$$
- Aleks can send **one kind of thing**:
  $$i \in \{0, 1\} := \text{public data}$$
  $$i \in \{0, 1, 2, 3\} := \text{classical data}$$
- ...and Bob gets **another kind of thing**:
  $$b := \text{private data}$$
  $$\psi := \text{quantum state}$$
Thanks!