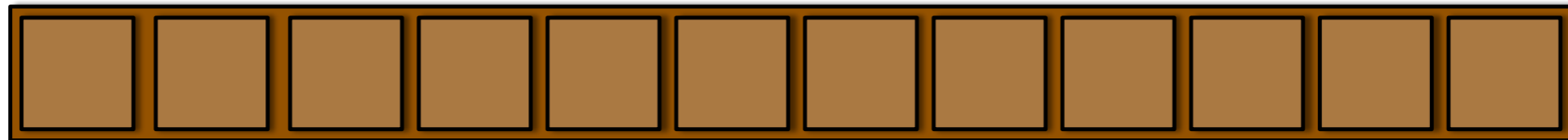
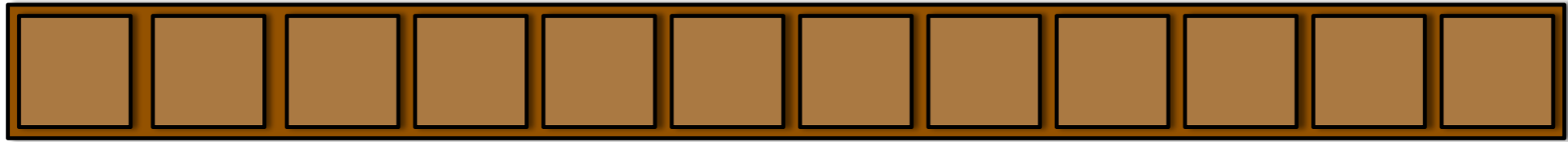


# The Chocolate Turing Machine

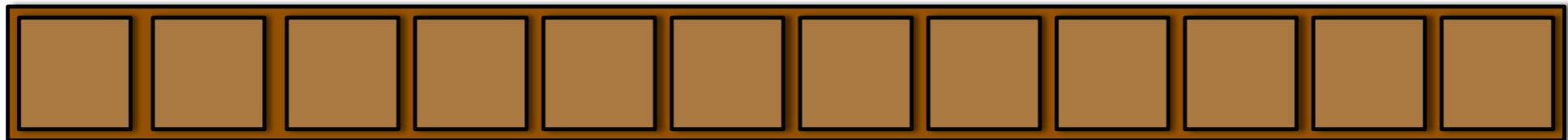


Paul Curzon  
Queen Mary University of London

**Twitter: @cs4fn**



# WHAT IS A TURING MACHINE?

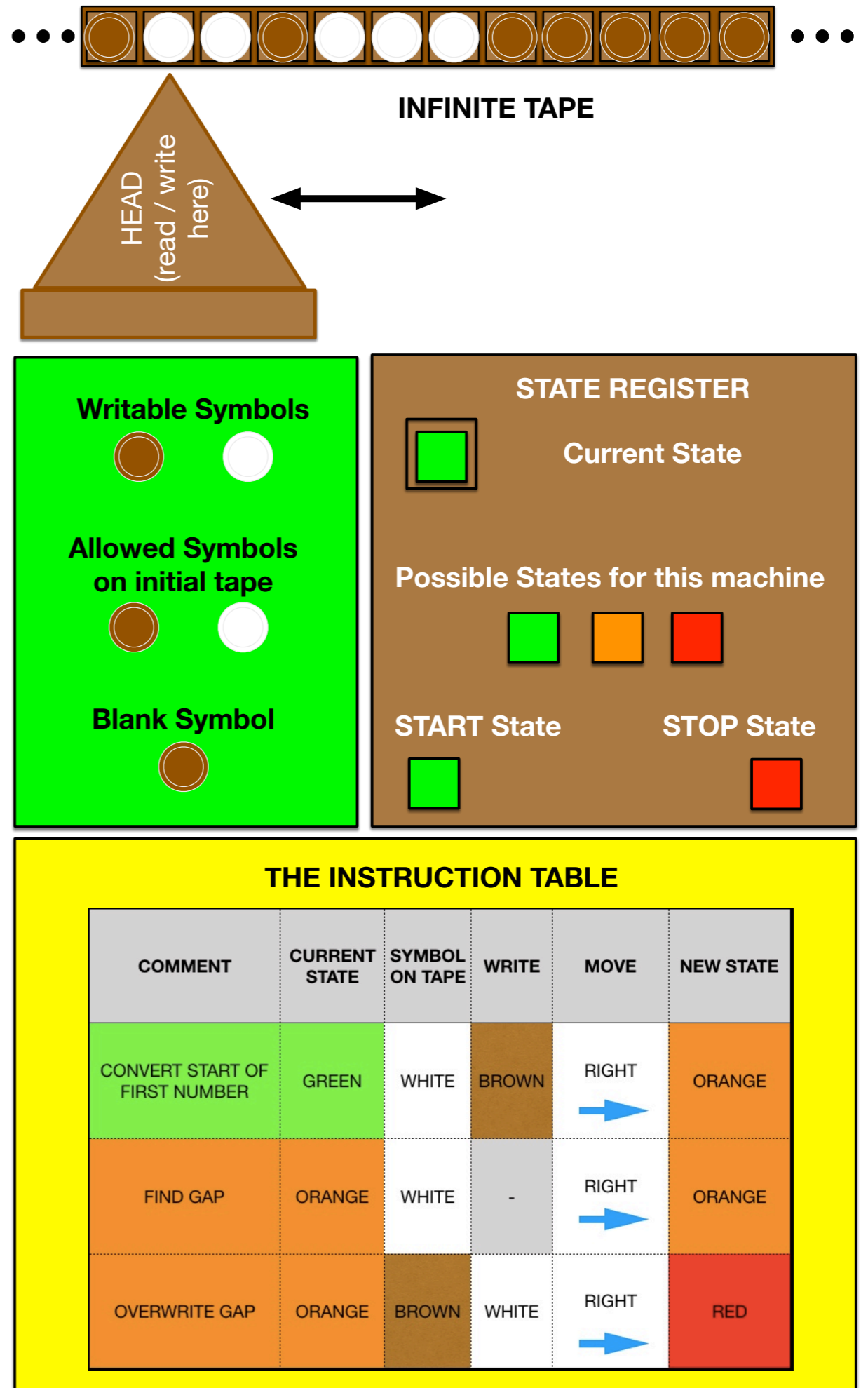


# A Turing Machine is a simple model of computation

- It **captures the essence of computation**, so what computers can do.
- A Turing machine can do any computation that any computer can do, now or in the future (just not as fast!)
- Created before a working computer existed
- Each Turing Machine does one fixed job like a gadget with an embedded, fixed program
- Turing imagined a human ‘computer’ slavishly working it

# What does a Turing Machine consist of?

# Make it from chocolate



# It consists of...

- An infinite tape (its **memory**) containing symbols
- Symbols that can be written on to the tape (eg 0, 1)
- Symbols that can appear on the original tape, including a BLANK symbol
- A read/write head (the one point it can change values)
- A finite set of possible states.
- A state register (to keep track of the state)
  - The current state from a fixed set (its “state of mind”)
  - A START state and a STOP state (if the machine enters this state it stops)
- A finite table of instructions (the **program**)

# Input-Output

You can think of a Turing Machine as following the input-process-output pattern

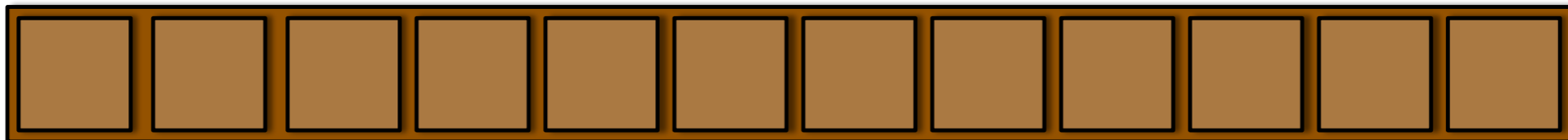
- **INPUT:** the initial tape
- **PROCESS:** the operation of the machine
- **OUTPUT:** the final tape once it **STOPS**

# Health Warning

- The details of how a Turing Machine is described can vary in lots of small ways that have no real bearing on what it does...
  - eg
    - the tape may be infinite in both directions OR just in one...
    - the tape may move OR the head may move
- **CHECK EXACTLY WHAT YOUR EXAM BOARD EXPECTS**



# IN PRACTICE





# Made of chocolate?

- It is a mathematical model but it can be thought of as an actual machine you can build.
- Computation can happen anywhere...not just on silicon
- Computation is about manipulating symbols
  - Symbols can be anything
- My version is made of chocolate...

# Number Representation

- We need a way to represent numbers
- We can use ANY representation we like
- To keep our first program simple we will use the earliest number representation used by ancient shepherds to count goats! UNARY
  - eg 5 is represented as ooooo
- A more sophisticated program could use binary, base 10 or even roman numerals - just different tape symbols.




# To ADD two unary numbers...

- eg with a starting tape:

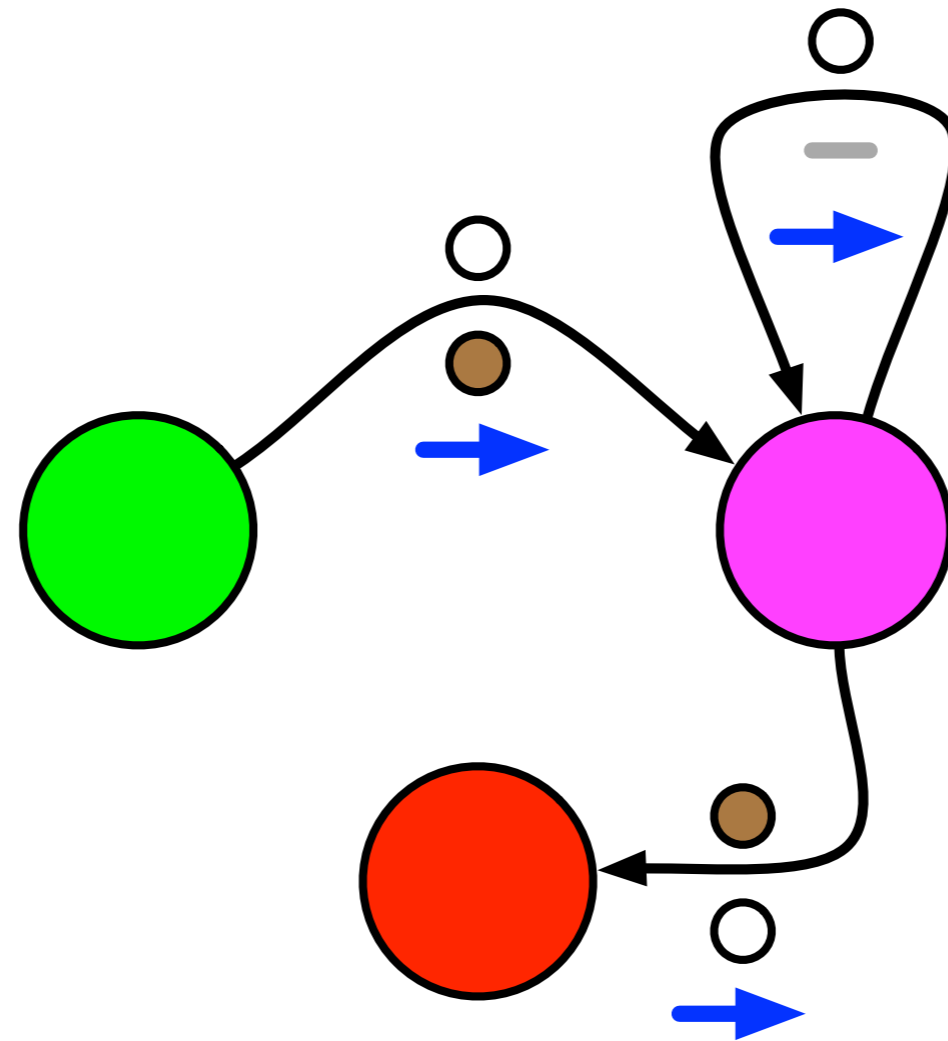
...\_00\_000\_ \_ \_ ...

1. Blank the first symbol (at start of first number)
  2. Write an o symbol in the gap between the numbers
- Assumes the numbers are separated by a single blank

# An Instruction Table

COMMENT	CURRENT STATE	SYMBOL ON TAPE	WRITE	MOVE	NEW STATE
CONVERT START OF FIRST NUMBER	GREEN	WHITE	BROWN	RIGHT 	ORANGE
FIND GAP	ORANGE	WHITE	-	RIGHT 	ORANGE
OVERWRITE GAP	ORANGE	BROWN	WHITE	RIGHT 	RED

# Instruction Tables can be thought of as State Transition Diagrams



# Make it from chocolate...

## Or role play it...

- Have one person for each tape position
  - eg holding a sheet with the symbol at that tape position
- Someone else is the tape head
- A final person acts as the state
  - different coloured hats for different states

# Another way

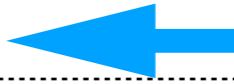
## To ADD two unary numbers

- This time leaving it in the starting from the same position
- eg with a starting tape:

...\_00\_000\_ \_ \_ ...

1. Blank the LAST symbol (at end of second number)
  2. Write an o symbol in the gap
- Assumes the numbers are separated by a single blank

# Another way to do the same thing: Adding to the other end of the number

COMMENT	CURRENT STATE	SYMBOL ON TAPE	WRITE	MOVE	NEW STATE
TRAVERSING FIRST NUMBER	GREEN	WHITE	-	RIGHT 	GREEN
	GREEN	BROWN	-	RIGHT 	PURPLE
TRAVERSING SECOND NUMBER	PURPLE	WHITE	-	RIGHT 	PURPLE
	PURPLE	BROWN	-	LEFT 	BLUE
CONVERTING END OF SECOND NUMBER	BLUE	-	BROWN	LEFT 	ORANGE
MOVING BACK TO GAP	ORANGE	WHITE	-	LEFT 	ORANGE
FILL GAP AND STOP	ORANGE	BROWN	WHITE	LEFT 	RED

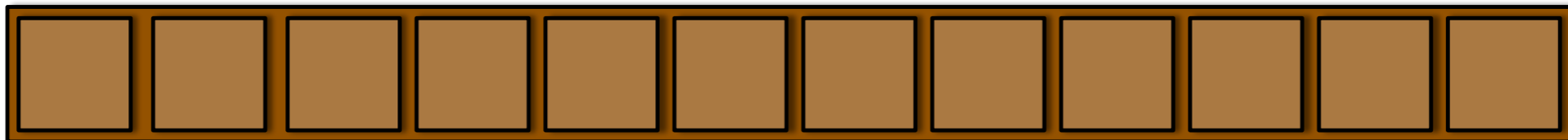


# Exercises




1. Work out a Turing Machine to Add 2 to a unary number
  - It should add it to the right-hand end of the number leaving the number starting in the same place
  - Extension ...
    - The Head should be left at the right hand end of the number, where it started.
2. Work out a Turing Machine to DOUBLE a binary number
  - The Machine can have three symbols (BLANK, 1 and 0)








# SOLUTIONS









# An Add 2 Instruction Table

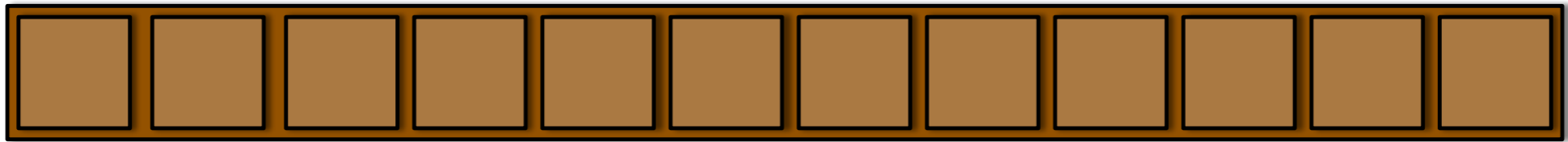
COMMENT	CURRENT STATE	SYMBOL ON TAPE	WRITE	MOVE	NEW STATE
MOVE TO END OF NUMBER	GREEN	WHITE	-	RIGHT 	GREEN
ADD 1	GREEN	BROWN	WHITE	RIGHT 	ORANGE
ADD ANOTHER 1	ORANGE	BROWN	WHITE	RIGHT 	RED

# Add 2, returning to the start

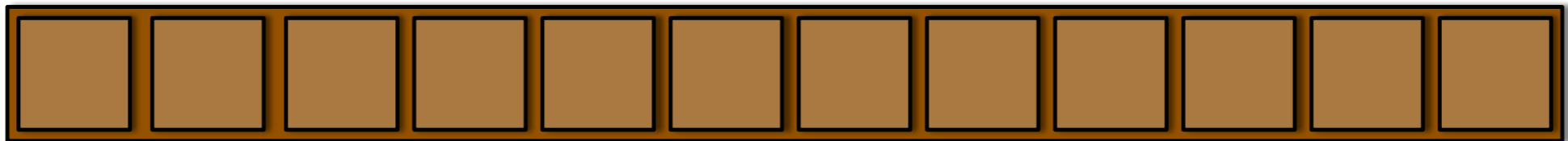
COMMENT	CURRENT STATE	SYMBOL ON TAPE	WRITE	MOVE	NEW STATE
MOVE TO END OF NUMBER	GREEN	WHITE	-	RIGHT 	GREEN
ADD 1	GREEN	BROWN	WHITE	RIGHT 	ORANGE
ADD ANOTHER 1	ORANGE	BROWN	WHITE	LEFT 	PURPLE
HEAD BACK TO START	PURPLE	WHITE	-	LEFT 	PURPLE
	PURPLE	BROWN	-	RIGHT 	RED

# Binary Double, returning to the start (just add a 0 on the end!)

COMMENT	CURRENT STATE	SYMBOL ON TAPE	WRITE	MOVE	NEW STATE
TRAVERSING NUMBER	GREEN	0	-	RIGHT 	GREEN
	GREEN	1	-	RIGHT 	GREEN
ADD A 0 AT THE START	GREEN	BLANK	0	LEFT 	ORANGE
RETURN TO START POSITION	ORANGE	0	-	LEFT 	ORANGE
	ORANGE	1	-	LEFT 	ORANGE
	ORANGE	BLANK	-	RIGHT 	RED



# THEORY



# As powerful as is possible

- Turing Machines can be programmed to do anything that any computer can do
  - They can compute any computation that it is possible for a computer to compute!
- Has **sequence, selection & repetition**
- You create a bespoke Turing Machine to do the computation you need to be done.

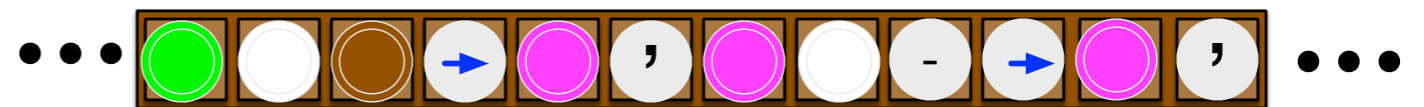
# Towards a Universal Turing Machine

- It is possible to write a program to simulate Turing Machines on a modern computer
  - NB the infinite tape is a problem though!
- Use that idea to see Turing's idea of a Universal Turing Machine
- You can create one Turing Machine (called U or UNIVERSAL) that can simulate any/all other Turing Machines
  - just need to write the simulation program



# A Universal Turing Machine

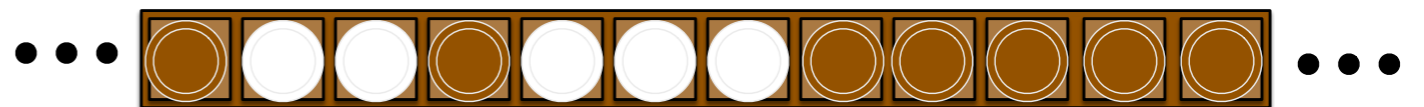
- Give UNIVERSAL the description of the Turing Machine you want to simulate as **data** on its tape



A REPRESENTATION OF  
THE INSTRUCTION TABLE



A REPRESENTATION OF  
THE CURRENT STATE

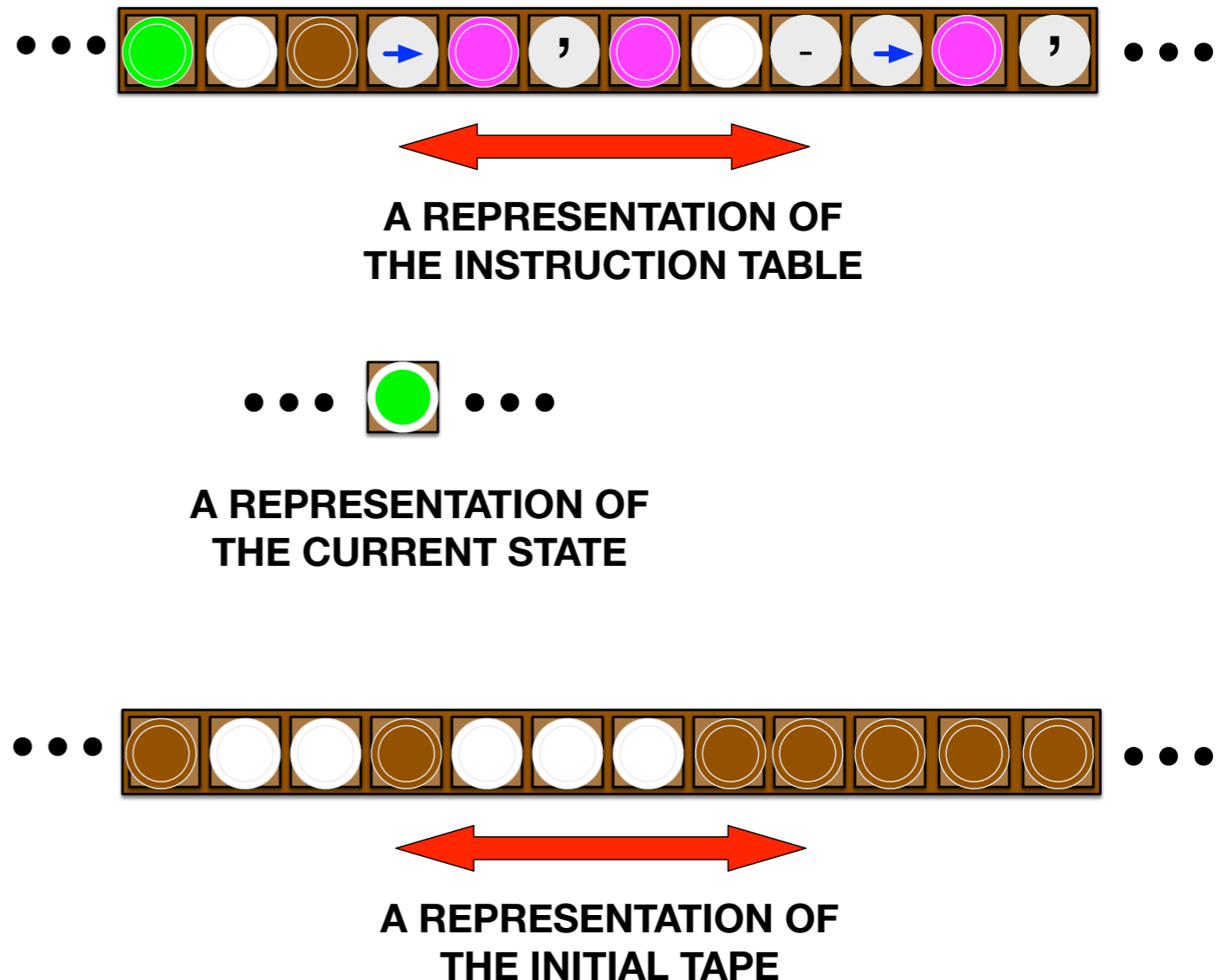


A REPRESENTATION OF  
THE INITIAL TAPE

- ... the instruction table (program), the initial state, data on initial tape etc in different areas of UNIVERSAL's tape

# A Universal Turing Machine

- UNIVERSAL's instructions tell it how to interpret that description (eg following a fetch-decode-execute cycle)
- UNIVERSAL's instructions change the areas of its tape (ie current state and data area) based on what it finds in the instruction table area



# A fundamental breakthrough

The idea of a Universal Turing Machine is the basis for a

- **general-purpose computer**
  - that can do anything
- because it is controlled by a **stored program** that tells it what to do.

***A fundamental theoretical breakthrough***

***in computer science***

# Summary

- A Turing Machine is just a simple model of what it is possible for computation to do
- Turing Machines can compute anything any real computer can compute
- You can make a UNIVERSAL Turing Machine that can simulate all other Turing Machines
  - The description of the Turing Machine is just the data on UNIVERSAL's initial tape

# Isaac Computer Science

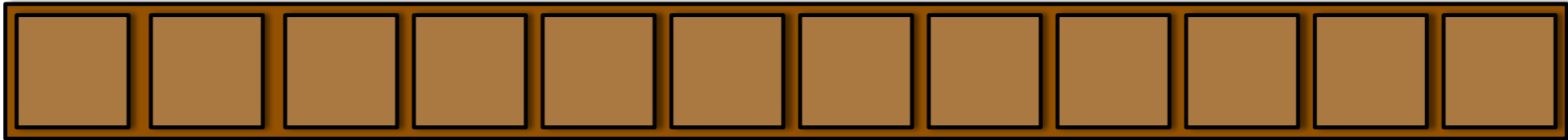
- Isaac Computer Science has detailed resources on Turing Machines

- See

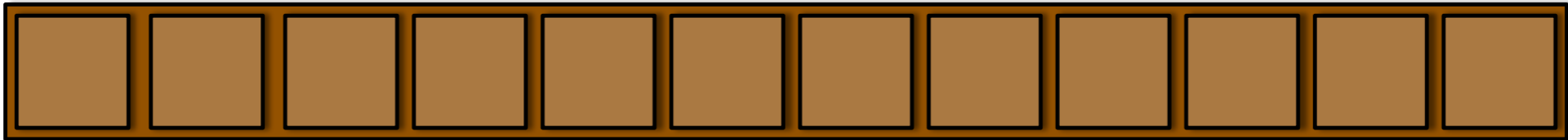
[isaaccomputerscience.org/topics/](https://isaaccomputerscience.org/topics/)

-> Theory of computation (AQA)

-> Turing Machines



# Thank You



## Our resources at

[teachinglondoncomputing.org/turingmachine/](https://teachinglondoncomputing.org/turingmachine/)

Twitter: @cs4fn

