The Church-Turing Thesis

What is an algorithm? "a rule for solving a mathematical problem in a finite number of steps"…Chambers' Dictionary

"process or rules for (esp. machine) calculation" Oxford dictionary

"Algorithm"..from al-Khwarazmi ("a native of Khwarazm") - the 9th Century mathematician Abu Ja'far Mohammed ben Musa

Algorithms pre-date Al-Khwarazmi..eg Euclid's Algorithm (gcf) Eratosthenes' Sieve (primes)

In the late 19th Century, a problem exercising mathematicians was one of those posed by Hilbert:

briefly.. "Is there a Universal Algorithm which can solve all Mathematical problems?"..attempts to find one failed..

..so perhaps there isn't one..

.. can we prove there is no universal algorithm?

.. we need to be able to define an algorithm precisely so as to prove properties of algorithms

a formalism of algorithms should be..

- precise and unambiguous
- simple
- general

Formalisms for Algorithms

By the 1930s the emphasis was on formalising algorithms

Alan Turing, at Cambridge, devised an abstract machine now called a Turing Machine to define/represent algorithms

Alonso Church, at Princeton, devised the Lambda Calculus which formalises algorithms as functions..more later in the course.

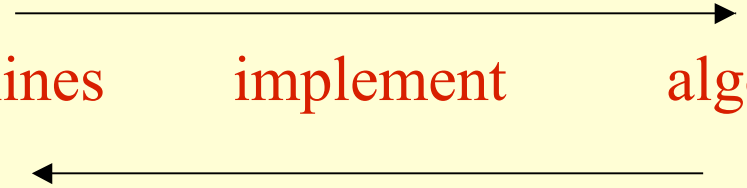
neither knew of the other's work in progress..both published in 1936

the demonstrated equivalence of their formalisms strengthened both their claims to validity, expressed as the Church-Turing Thesis..

Turing Machines: precise
simple
general ?

the Church-Turing Thesis:

"a problem can be solved by an algorithm iff it can be solved by a Turing Machine"



all algorithmically solvable problems can be solved by a Turing Machine

"a function is computable iff it can be solved by a Turing Machine"

" an algorithm is what a Turing Machine implements"

Thesis not Theorem:

because we cannot prove this..
with a counter example we could disprove it
(but this has not been done).

we can show supporting evidence for the validity of the thesis

the Church-Turing Thesis: types of evidence

- large sets of Turing-Computable functions

  many examples…no counter-examples

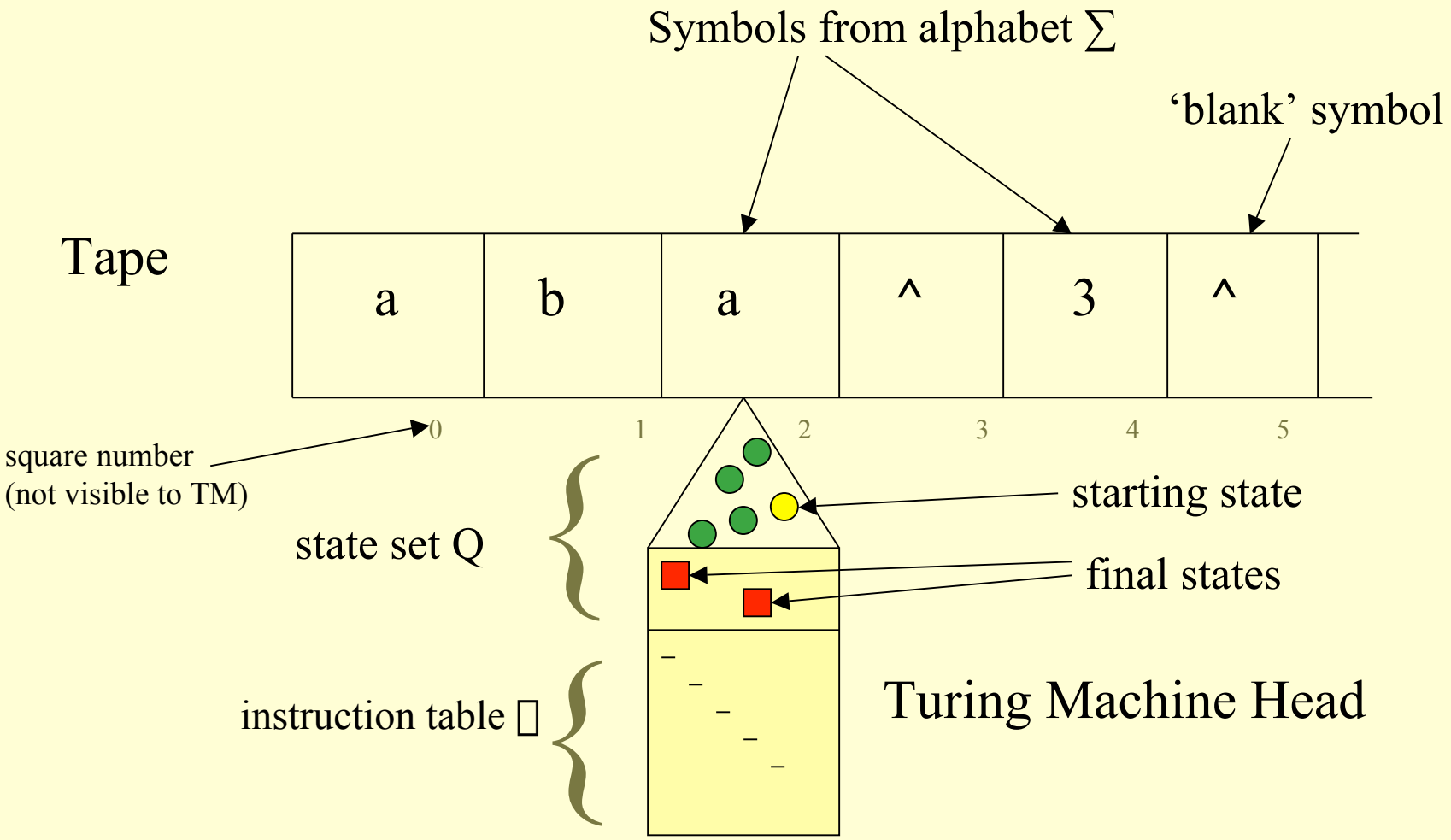
- equivalent to other formalisms for algorithms

  Church's □ calculus and others

- intuitive - any detailed algorithm for manual calculation can be implemented by a Turing Machine.

  via Turing Machine implementation of mechanical methods

A Turing Machine



A Turing Machine Run

Start:

- in initial state
- head over square 0
- finite number of non-blank symbols at start of tape
  rest of tape blank - contains ^

A run is a step-by-step computation:

- reads symbol on current square
- writes a symbol from alphabet $\sum$ to current square
- moves left 1 square, right 1 square or does not move
- enters a new state

..according to…

..the Instruction Table:

depending on
- current state
- symbol in current square

the table gives
- symbol to write
- direction to move
- new state

the **Instruction Table**, δ, is the **program** of the Turing Machine also called the **δ-function**:

δ(current-state,current-symbol) = (new-state, new-symbol, move)

Stop: the run stops when..

a) it reaches a final, or halting state:

- the TM stops (halts) and succeeds.
- the output is the tape contents from square 0 up to (but not including) the first ^

or b) the pair (current-state,current-symbol) is not in the instruction table ("no applicable instruction"):

- the TM halts and fails
- the output is undefined

or c) the head tries to move left from square 0:

- the TM halts and fails
- the output is undefined

OR..the TM may not halt..it loops or runs forever

A Turing Machine