3004
Computability and Complexity Theory

Fabio Zanasi
http://www0.cs.ucl.ac.uk/staff/F.Zanasi/

Lecture two

Previously on COMP3004

What is a computer?

Are there limits to what a computer can do?

In this lecture

We begin the study of computability theory by introducing our first abstract model of computation: the **Turing Machine** (TM).

Towards Turing Machines

Alan Turing (1912-1954)

Known to the general public for deciphering the *Enigma* code in WW II

But before the war (1936) he wrote the paper:

On Computable Numbers,
with an Application to
the *Entscheidungsproblem*

Theoretical computer science was born.

Turing's question

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. Turing.

[Received 28 May, 1936.—Read 12 November, 1936.]

The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions

Turing's article posed the question:

what is computation?

Turing's Approach

Watch a human being calculating something.

- (S)he is following a set of rules
- symbols are read and written (say on paper)
- human behaviour changes depending on the symbol under examination.

Turing's Approach

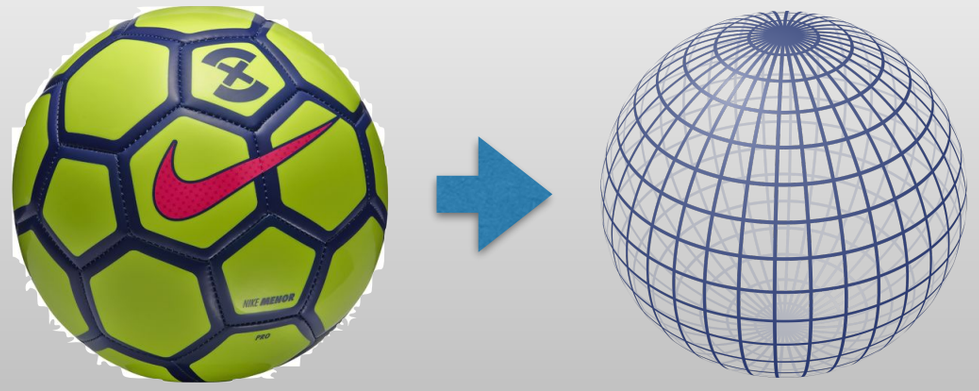
Task of the scientist (Turing):

extract from this process what is **essential**
and eliminate what is **irrelevant**.

Remember Aristotle

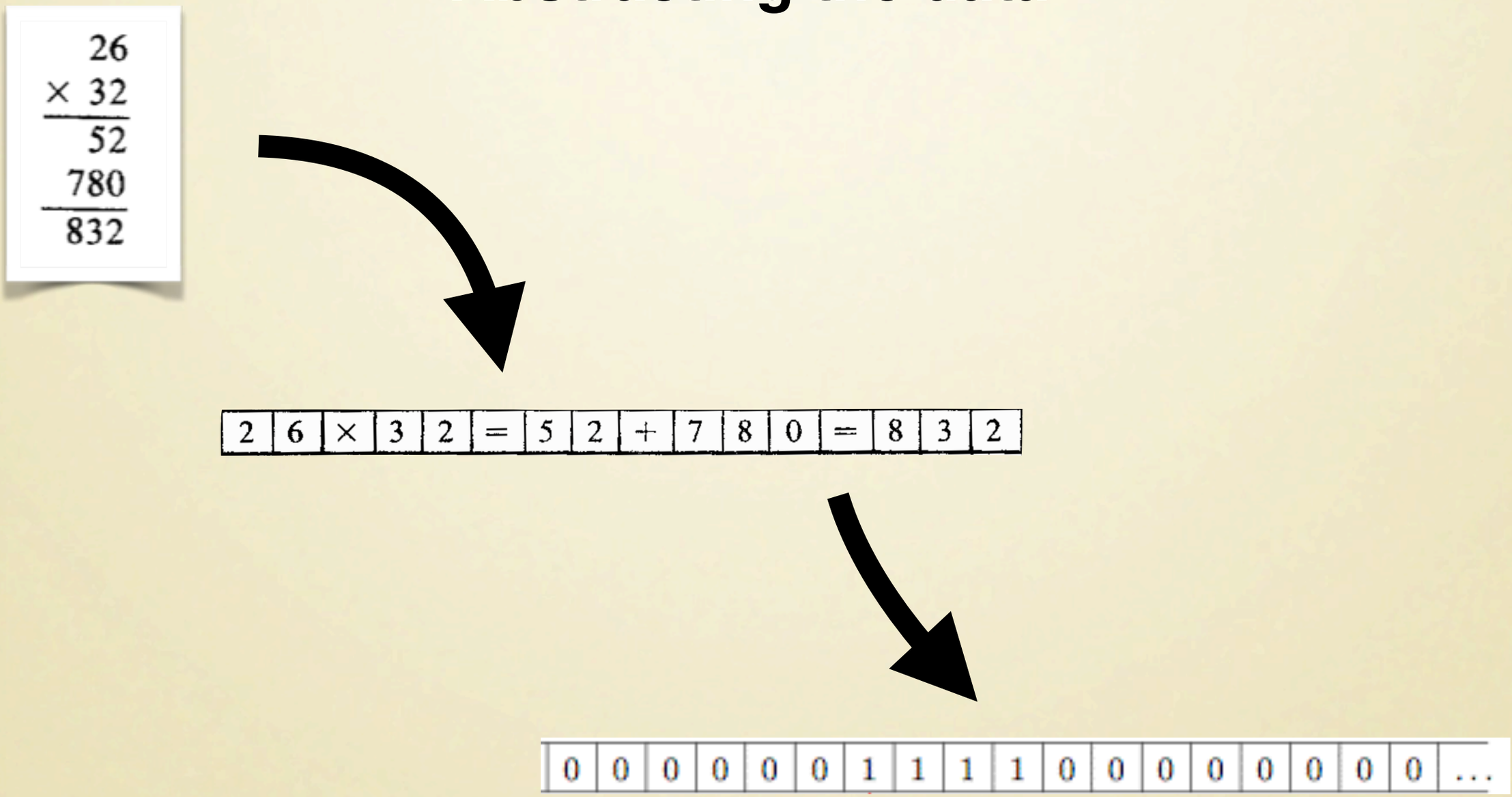
Essential vs accidental

Remember geometry



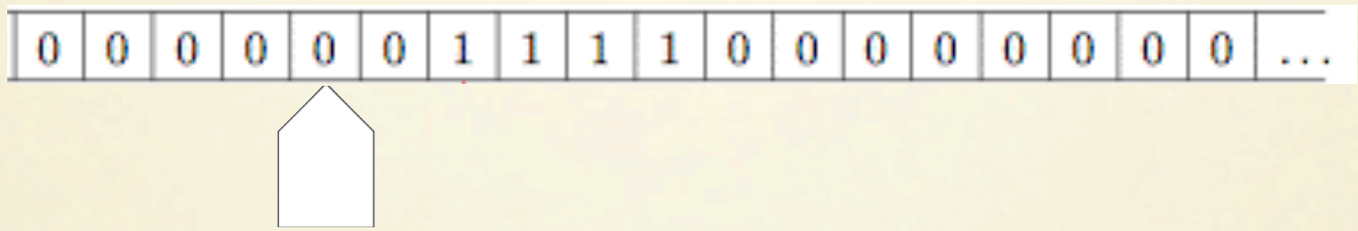
What is essential?

Abstracting the data



What is essential?

Abstracting the actions

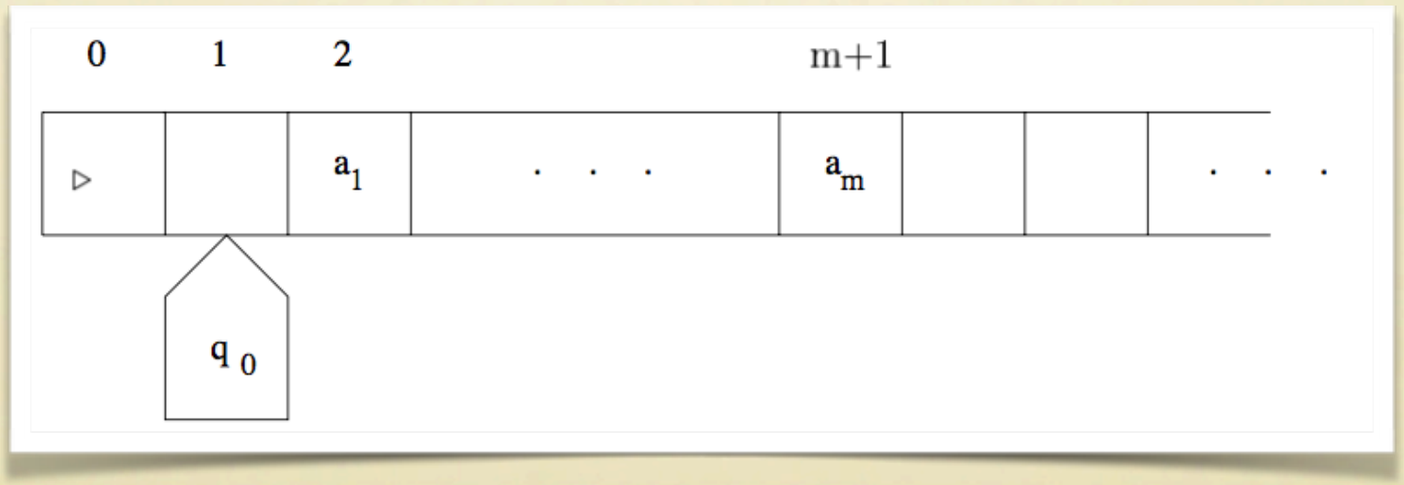


- Write the symbol 0
- Write the symbol 1
- Move one square to the right
- Move one square to the left
- Observe the symbol currently scanned and choose the next step accordingly
- Stop

The Turing Machine

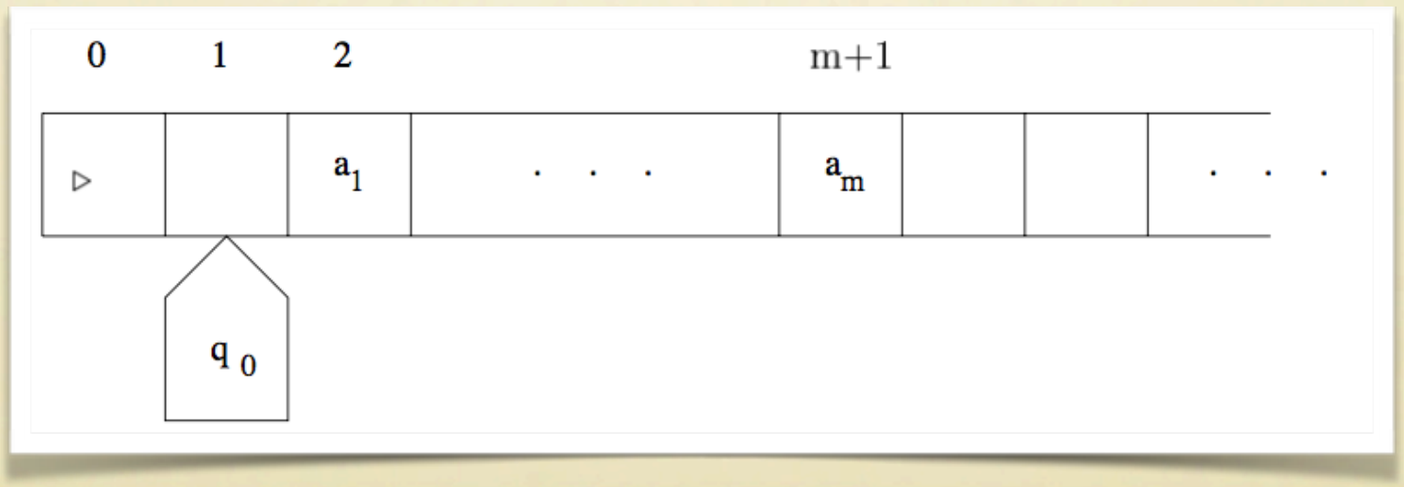
Turing Machine, informal definition

- A Turing Machine has a single **tape**. The tape has a left end and is infinite to the right. It is divided into numbered cells.
- The leftmost cell is numbered 0 and is always marked with the symbol ▷.
- Each cell on the tape can contain a symbol or be blank. Initially the tape is blank, with the exception that cell #0 contains ▷ and any **input string** $a = a_1 a_2 \ldots a_m$ is written in cells #2 to #m+1.
- A Turing Machine has a **head**, which during the computation can be in a finite number of **states**.
- Initially the head is in a distinguished starting state q_0 and scanning cell #1.



Turing Machine, informal definition

- At any time the head is **reading** the contents of a single cell on the tape. Based on the current content of the cell and the current state, the Turing Machine can do one of the following:
  - **Halt**.
  - Change to a new state and *either* **write** a new symbol in the current cell *or* **move** one cell to the left or right.
- There is an exception: if the head is reading the special symbol ▷ (left end of the tape), then it can only move to the right.
- The actions to be taken are specified by a **program**.



Turing Machine, formal definition

A Turing Machine is a tuple $\langle \Sigma, Q, q_0, H, \delta \rangle$

- Σ is a finite **alphabet** of symbols. It includes:
  - the blank symbol $\sqcup$
  - the end of tape symbol $\triangleright$
- Q is a finite set of **states**.
- $q_0 \in Q$ is the **initial state**.
- $H \subseteq Q$ is the set of **halting states**.
- δ is the **transition function**

$$\delta : \quad (Q \backslash H) \times \Sigma \quad \rightarrow \quad Q \times (\Sigma \uplus \{\rightarrow, \leftarrow\})$$

The transition function

$$\delta : \quad (Q \backslash H) \times \Sigma \quad \rightarrow \quad Q \times (\Sigma \uplus \{\rightarrow, \leftarrow\})$$

current state

symbol being scanned

next state

action: write symbol *or* move left *or* move right

- δ encapsulates the program governing the operation of the TM.
- δ is a *total* function.
- We can write the definition of δ as a set of 4-tuples:

$$\{\langle\, q_i,\, a,\, q_j,\, \sqcup\,\rangle, \langle\, q_k,\, \triangleright,\, q_l,\, \rightarrow\rangle, \langle\, q_i,\, b,\, q_k,\, \leftarrow\rangle, \ldots\}$$

The transition function

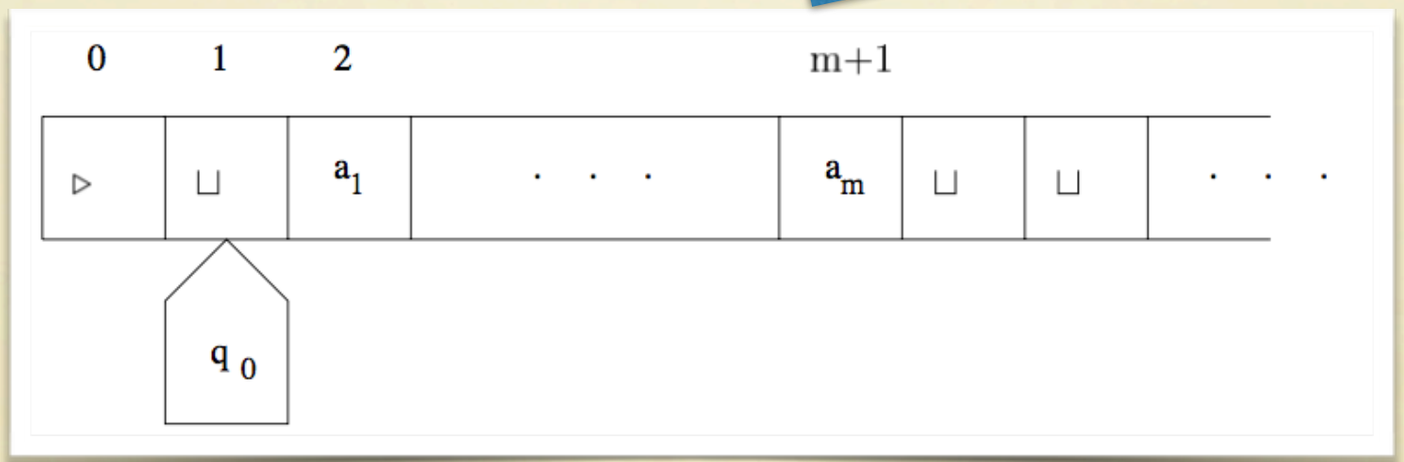
$$\delta : \quad (Q \backslash H) \times \Sigma \quad \rightarrow \quad Q \times (\Sigma \uplus \{\rightarrow, \leftarrow\})$$

Two caveats on the definition of δ:

If $\delta(q, \triangleright) = (q', a)$ *then* $a = \rightarrow$

If $\delta(q, a) = (q', b)$ *then* $b \neq \triangleright$

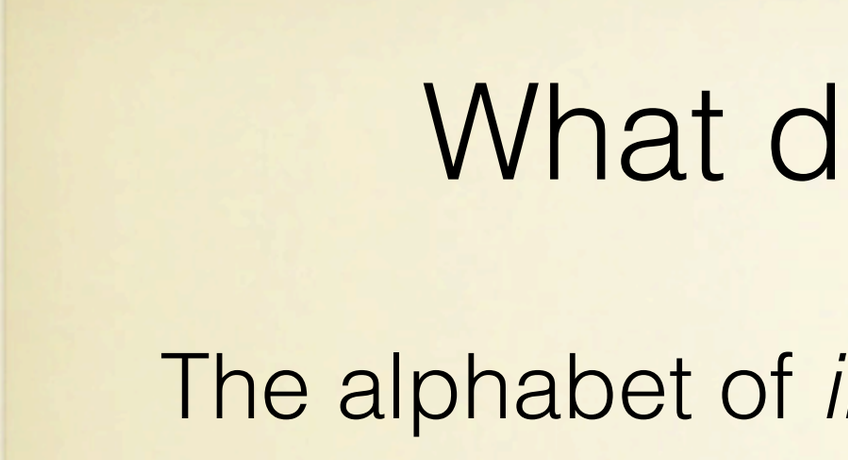
Exercise: why these restrictions?

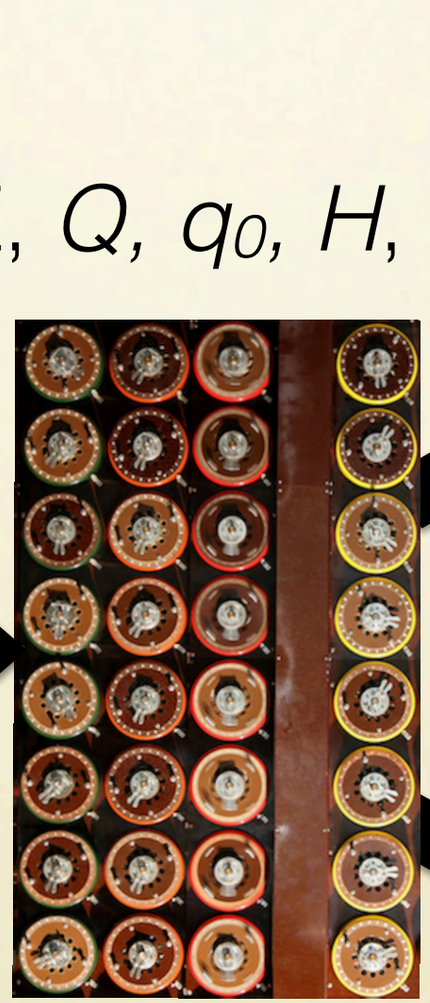


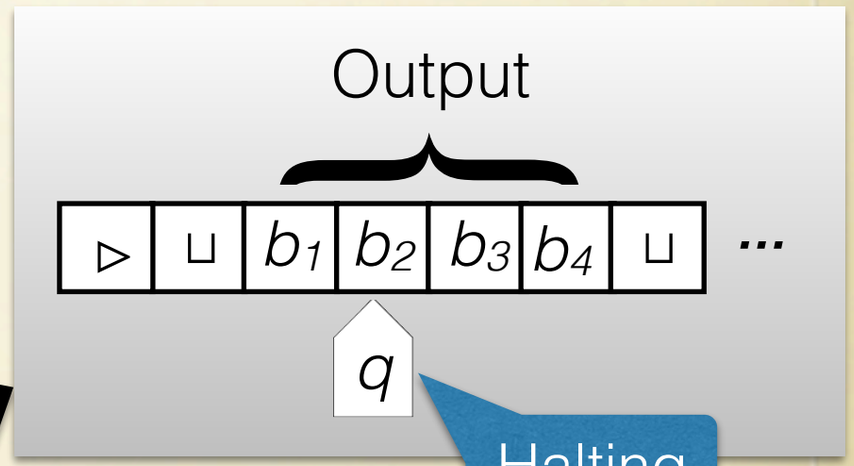
What does a TM compute?

The alphabet of *input/output symbols* Σ_I is $\Sigma \backslash \{\sqcup, \triangleright\}$.

$\langle\, \Sigma,\ Q,\ q_0,\ H,\ \delta \,\rangle$







A toy example

$\langle\ \Sigma,\ Q,\ q_0,\ H,\ \delta\ \rangle$

$\Sigma = \{1, \triangleright, \sqcup\}$

$\Sigma_I = \{1\}$

$Q = \{q_0, q_1, q_2, h\}$

$H = \{h\}$

$\delta(q_0,\sqcup) = (q_1, \rightarrow)$ $\quad$ $\delta(q_0,1) = (q_1, \rightarrow)$

$\delta(q_1,\sqcup) = (q_2, 1)$ $\quad$ $\delta(q_1,1) = (q_1, \rightarrow)$

$\delta(q_2,\sqcup) = (h, \sqcup)$ $\quad$ $\delta(q_2,1) = (q_2, \leftarrow)$

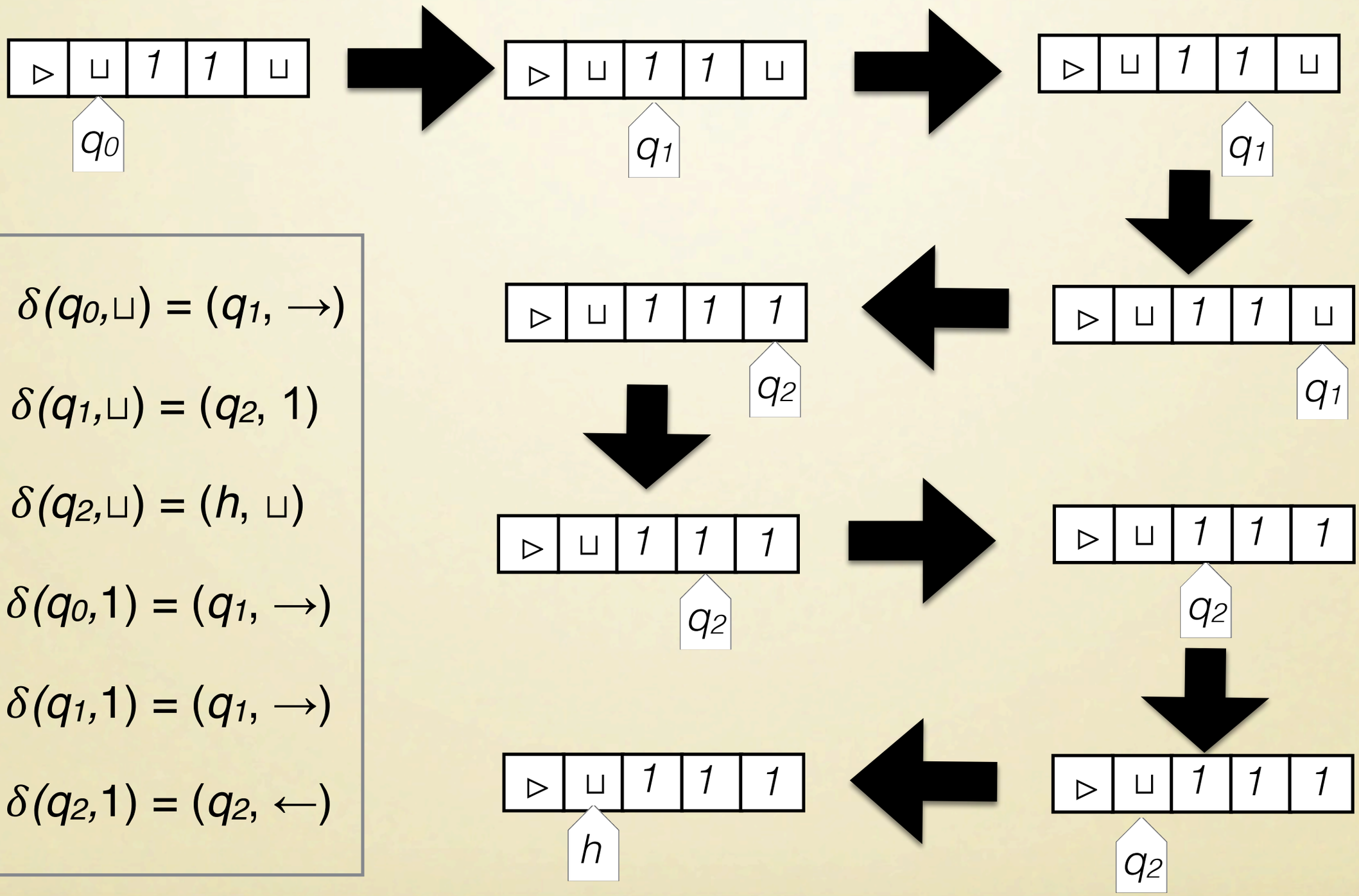
$\delta(q, \triangleright)$ = whatever for all $q \in Q \setminus H$

Exercise

What does this machine compute?

Try it on

A toy example



$\delta(q_0,\sqcup) = (q_1, \rightarrow)$

$\delta(q_1,\sqcup) = (q_2, 1)$

$\delta(q_2,\sqcup) = (h, \sqcup)$

$\delta(q_0,1) = (q_1, \rightarrow)$

$\delta(q_1,1) = (q_1, \rightarrow)$

$\delta(q_2,1) = (q_2, \leftarrow)$

Diagrammatic Representation of Turing Machines

The "add 1" Turing Machine

$\Sigma = \{1, \triangleright, \sqcup\}$

$\Sigma_I = \{1\}$

$Q = \{q_0, q_1, q_2, h\}$

$H = \{h\}$

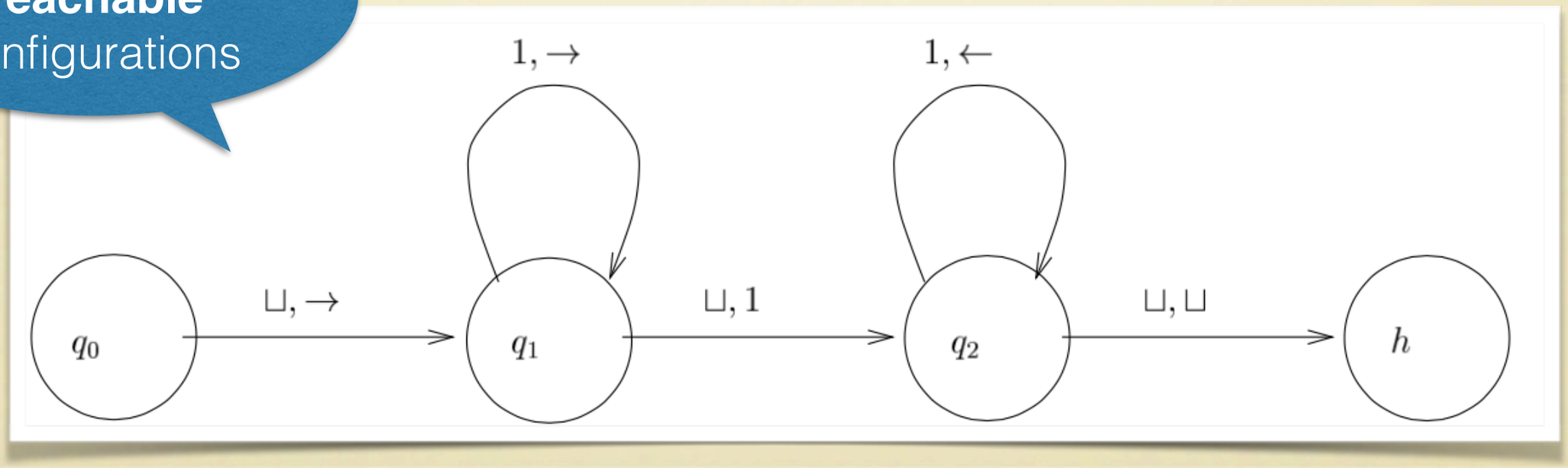
$\delta(q_0,\sqcup) = (q_1, \rightarrow)$ $\quad$ $\delta(q_0,1) = (q_1, \rightarrow)$

$\delta(q_1,\sqcup) = (q_2, 1)$ $\quad$ $\delta(q_1,1) = (q_1, \rightarrow)$

$\delta(q_2,\sqcup) = (h, \sqcup)$ $\quad$ $\delta(q_2,1) = (q_2, \leftarrow)$

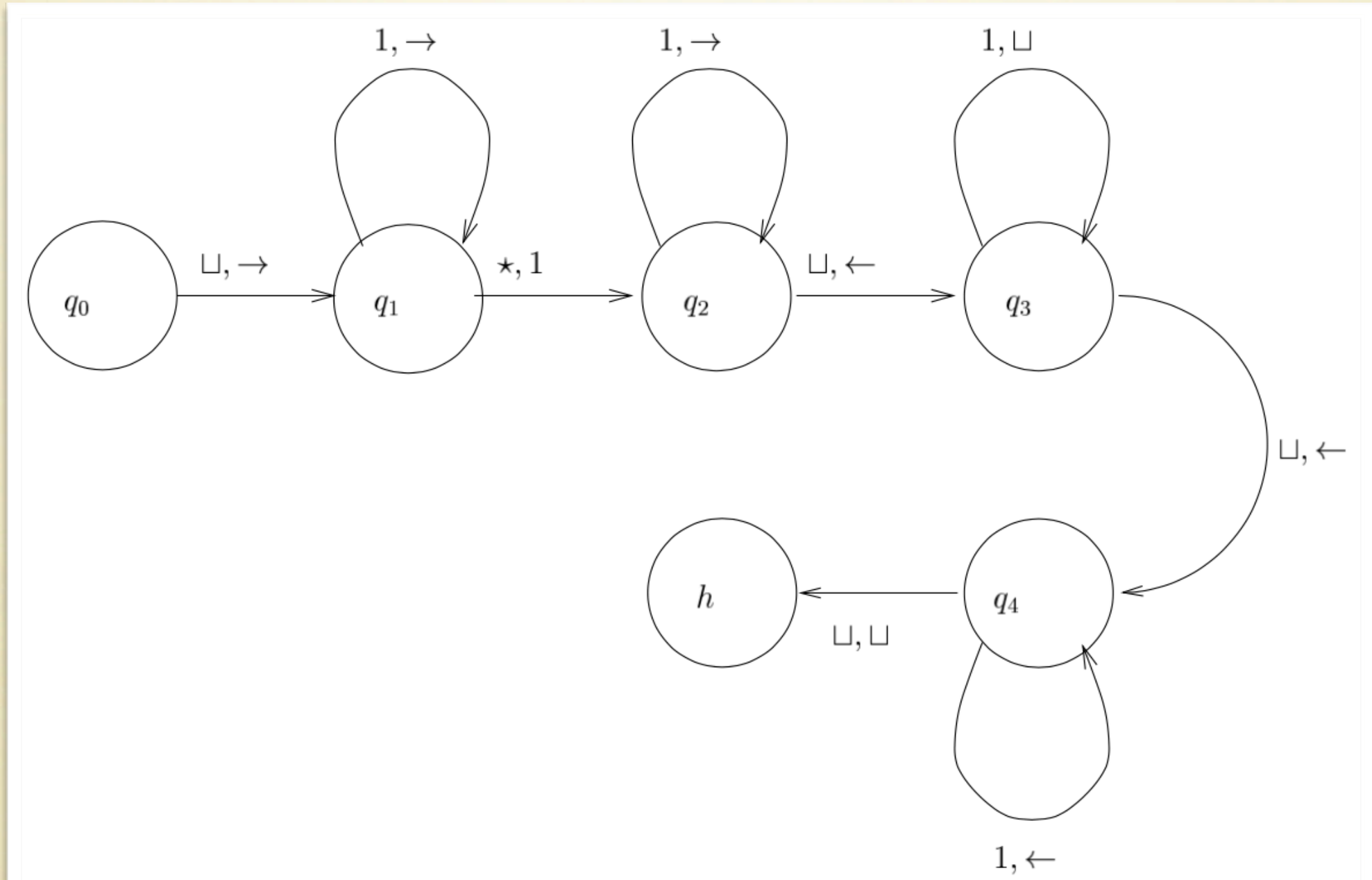
$\delta(q, \triangleright) =$ whatever for all $q \in Q \setminus H$

We only draw **reachable** configurations



Example: general unary addition

$\Sigma = \{1, \bigstar, \triangleright, \sqcup\}$ $\Sigma_I = \{1, \bigstar\}$ $Q = \{q_0, q_1, q_2, q_3, q_4, h\}$ $H = \{h\}$

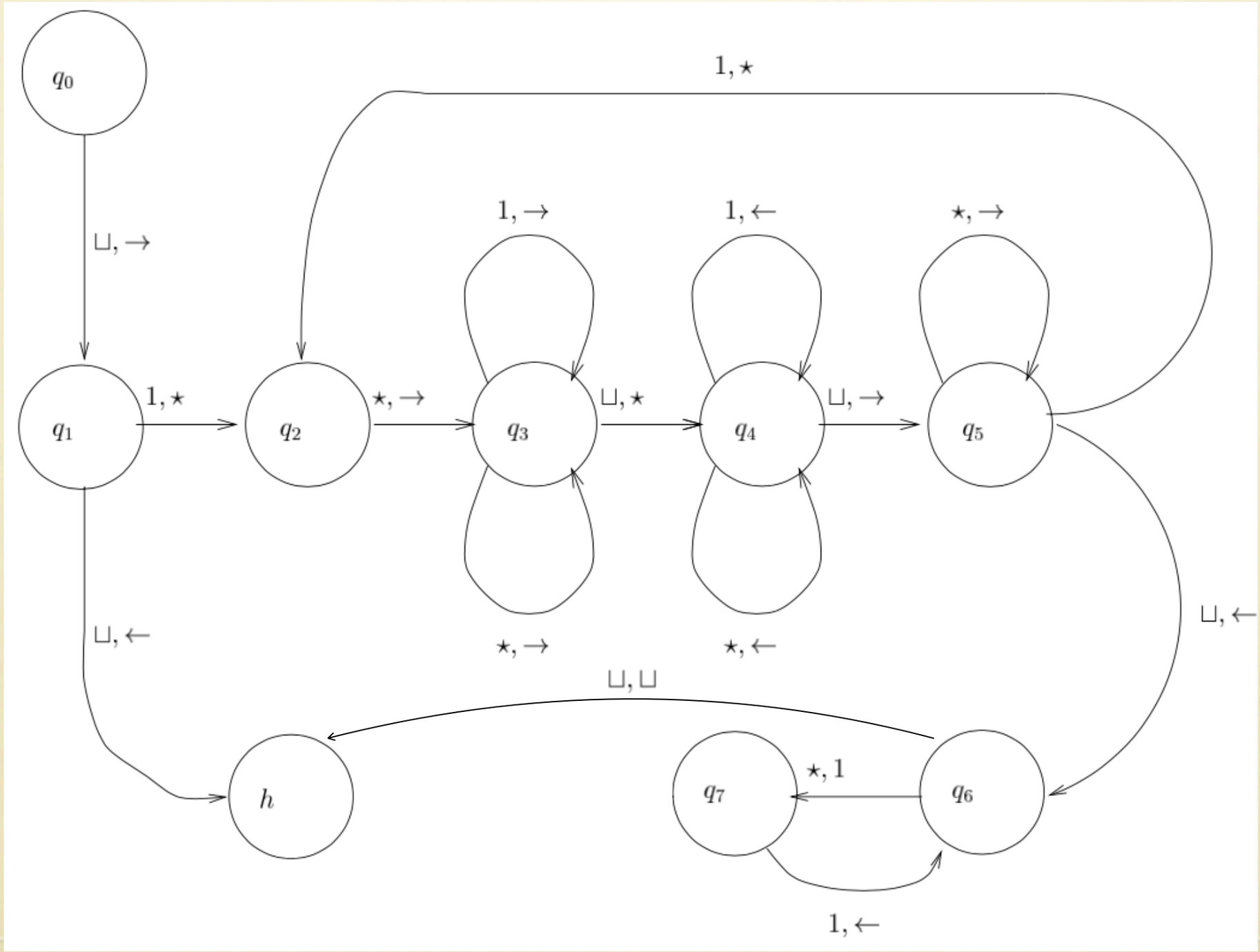


Computing functions $\mathbb{N}^k \to \mathbb{N}$

- What we have seen so far are Turing machines computing functions from $\mathbb{N}^k$ to $\mathbb{N}$.
- We can do that in general:
  - Numbers are represented in unary notation.
  - A k-tuple $x \in \mathbb{N}^k$ can be represented using unary notation numbers separated by the ★ symbol.
    - Example: $(4,5,1) \in \mathbb{N}^3$ is encoded as 1111★11111★1.
- Thus the input alphabet is going to be $\Sigma_I = \{1,★\}$.

Example: unary multiplication by 2

$\Sigma = \{1, \bigstar, \triangleright, \sqcup\}$ $\Sigma_I = \{1\}$ $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, h\}$ $H = \{h\}$



Coda

“Forgotten” by History

Emil Post (1897-1954)

He anticipated many of the insights of Turing (and others, including Gödel), but his work remained unpublished for many years.