# Universal Turing Machines

So far,

**different problems ⇒ different Turing machines..**

a TM has been single-algorithm special purpose hardware.

Turing also defined **U**, the **Universal Turing Machine**

- an ordinary TM
- which **calculates $f_M(w)$ for a TM, M running on input w**

We give U a description of M and the input word w.

U calculates what M would do

- it is an interpreter for arbitrary TMs.

## **Universal Turing Machines…**continued

necessary conditions:

- the input alphabet of M must not contain symbols absent from the input alphabet of U
- the output alphabet of M must not contain symbols absent from the output alphabet of U

So we build U for **Standard Turing Machines** with a fixed alphabet which is also the alphabet of U.

Our fixed alphabet, **C** is the typical typewriter alphabet without the blank symbol,^:

**C = {a,b,c,d,e,f,g,..y,z,A,B,C,D…,Z,1,2,3,4..,9,+_)(*,&,%,$,£,@…}**

# Definition of a **Standard Turing Machine**

1. let C be the alphabet {a,b,c,..A,B,C…1,2,3..!@£,,,}
   (the typewriter alphabet without ^)

2. a Turing Machine S is said to be Standard if it
   - is a 6-tuple $(Q, \sum, I, q_0, \delta, F)$ with
     - Q finite set of states
     - $\delta$ a partial function $\delta: Q \times \sum \Rightarrow Q \times \sum \times \{-1,0,1\}$
     - $F \subset Q$, halting states
     - $q_0$ starting state
     - $I = C$
     - $\sum = C \cup \{\wedge\}$

3. $\Rightarrow$ S has
   - a single 1-way infinite tape
   - one-track tape
   - marking of square 0 must be explicit - no alphabet extension.

## Coding a standard Turing Machine, S

Q is finite: label the states with integers 0, 1, …n., $q_0 \equiv 0$,
with the halting states f, f+1,…n for some $0 < f \leq n$.

The description of S to be given to U must use only the alphabet C.

**S = (Q, C∪{^}, C, $q_0$, δ, F).**

Suppose Q = {0,1,2,3..n}, $q_0 = 0$

F = { f, f+1, ..n} $n \geq 0, f \leq n$

the δ-function entries: δ(q, s) = (q', s', d) where

s,s' ∈ C ∪ {^}, $0 \leq q < f$,

d∈ {-1,0,1}, $0 \leq q' \leq n$

We represent a δ-function entry by the 5-tuple (q, s, q',s',d)

# ..code of a standard TM, S..

General form of the code:

$n, f, t_1, t_2, t_3, \ldots, t_N$ a word in C ∪ {^}

**the 5-tuples (q, s, q', s', d)**

replace all ^ by 'blank' in the code ⇒ code(S) ∈ C*

So **code(S) = n, f, (q, s, q', s', d),…(q, s, q', s', d)**

where n, f, q, q', d are decimal numbers

$0 \leq f, q < f, q' \leq n$

$d \in \{-1, 0, 1\}$

s, s' ∈ C or = 'blank'

## which words of C code a TM?

eg “::(*)165ase?   2,2,(1,a,2,blank,-1)?

- ordering of instructions in the code doesn’t matter
- numbering of states..not a restriction
- Q is finite..so its members can be listed with the initial state always 0

code(S) has some redundancy..it is not unique for S..
..variation in:

.. allocation of numbers to states
.. permutation of the f-1 non-starting, non-halting states
.. permutation of the n-f+1 halting states.
.. permutation of the δ-function entries

## Building the Universal Turing Machine, U

For any standard TM, S, and any word w of C, we require

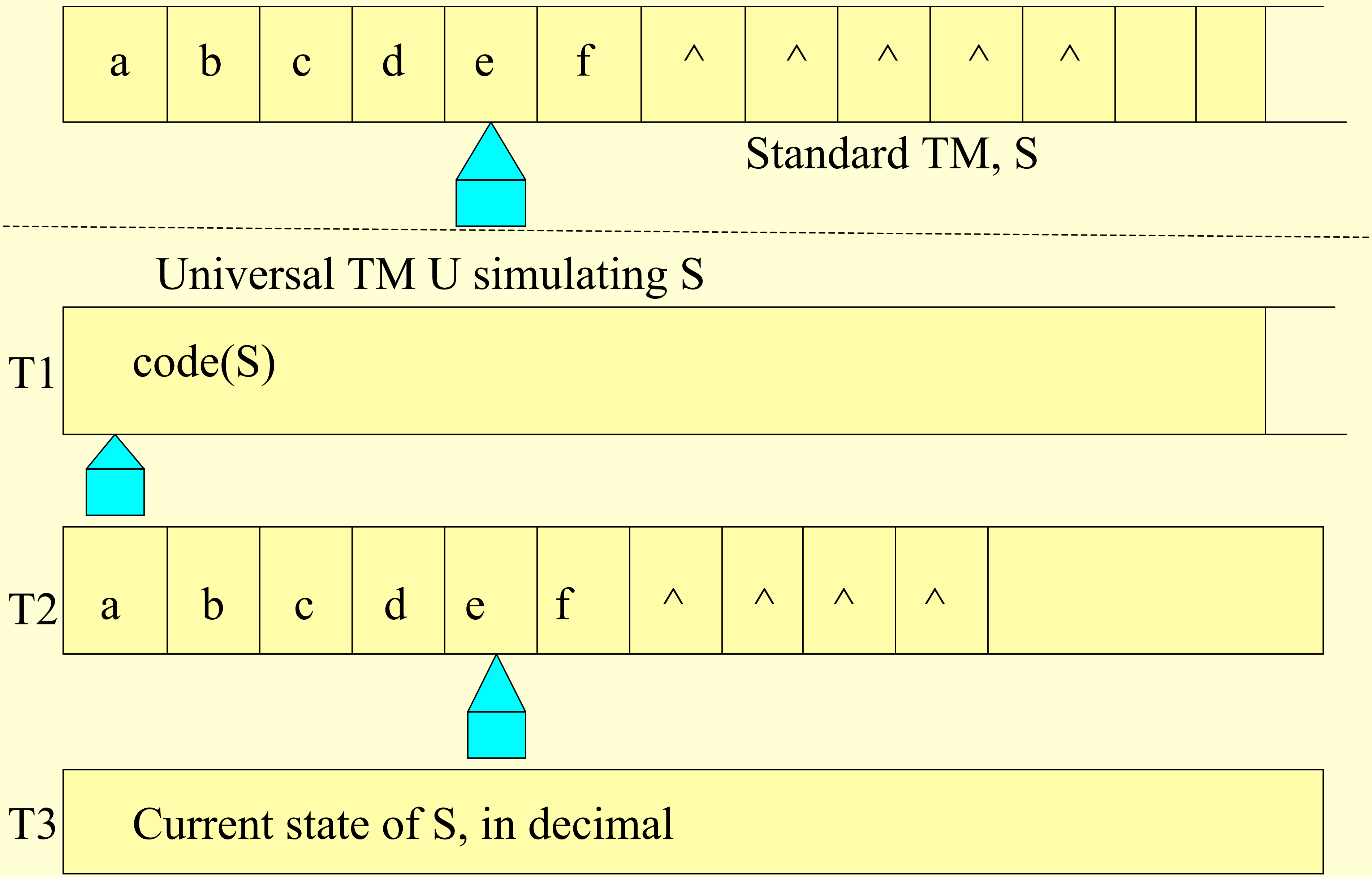
$$f_U\ (code(S) * w) = f_S\ (w)$$

U has input alphabet C, full alphabet C ∪ {^}

U simulates S, using code(S).

U has 3 tapes:

| | |
|---|---|
| Tape1 of U | contains code(S) |
| Tape 2 of U | same as the single tape of S |
| Tape 3 of U | contains the current state of S |


a
b
c
d
e
f
^
^
^
^
^
Standard TM, S
Universal TM U simulating S
T1
code(S)
T2
a
b
c
d
e
f
^
^
^
^
T3
Current state of S, in decimal

## the Operation of U

1. Initialise: U writes 0 in square 0 of Tape 3
   U copies w from tape1 to Tape 2
   U returns all 3 heads to square 0.

2. Simulation of S, current state q, for each step of execution
   - if $q \geq f \Rightarrow$ halting state..if so, output of S is on Tape 2 of U.
     U copies this to Tape 1, then ^.
     U Halts & Succeeds with S's output on Tape 1.

   - if $q < f \Rightarrow$ not a halting state
     U scans code(S) on Tape 1 for (q, s, q',s', d)
     where s is the current symbol on tape 2 (tape of S)
     if no (q, s, q', s',d)$\Rightarrow$no applicable instruction
     $\Rightarrow$ U moves left repeatedly.. Halt & Fail.
     otherwise ∃(q,s,q',s',d) on Tape1 in code(S) then
     simulate actions of S…

(..actions of S: S writes s' on its tape
S Head moves d
S goes into state q' )

so U: writes s' on Tape 2 (copy s' from tape 1)
writes q' on Tape 3 (copy q' from tape 1)

Head 1 returns to square 0
Head 2 moves d
Head 3 returns to square 0

. …end of cycle for (state q, current-symbol). This is now repeated.

What if S is non-standard with respect to its
input alphabet I,
or whole alphabet ∑?

**elimination of Scratch Characters** or

**"alphabet C is always enough"**

**Let** M = (Q, ∑, C, $q_0$, δ, F) with $f_M$: C* ⇒ C*

ie. input and output are both words of C, full alphabet ∑.

∑ includes C and scratch characters, so M is not standard:

**…then** there is a standard TM S equivalent to M..

S will use encoded characters to mimic M

…we need a code: ∑⇒ C* to represent symbols of ∑ as words of C

∑⊇ C, so w ∈ C* ⇒ w ∈ ∑*

a standard TM S first encodes $w_1$,$w_2$..$w_n$ (all are in C) giving code(w).

S uses codes throughout and simulates the actions of M.

if M Halts ⇒S decodes tape contents giving output; only chars in C ∪ {^}.

S simulates M, giving same output, so M and S are equivalent

Now U can interpret any TM M with $f_M$:C*⇒C*.

**U operates on code(S).**

# ..coding whole alphabet ∑ where ∑ ⊃ C?

∑ may have symbols not in C, but must be finite

…we can code a finite alphabet ∑ in C:

find an integer **k** such that

**no.words of C of length k = $88^k$ ≥ size of ∑.**

map the symbols of ∑ onto words of C of length k.

ie. 1-1 function code:∑⇒ $C^k$

for w = $a_1a_2a_3..a_n$ of ∑*

code(w)=code($a_1a_2a_3..a_n$) = code($a_1$).code($a_2$)..code($a_n$), which is a word of C, of length kn

decode:C*⇒∑*

such that decode(code(w)) = w, w ∈ ∑*, otherwise undefined.

# The code of the Tail TM

$Q = \{q_0, q_1, q_2, q_3, q_4\, q_5\}$  $F = \{q_5\} \Rightarrow n = 5, f = 5$

$\delta(q_0,a) = (q_1,\hat{},1)$
$\delta(q_0,b) = (q_1,\hat{},1)$

$\delta(q_1,a) = (q_1,a,1)$
$\delta(q_1,b) = (q_1,b,1)$
$\delta(q_1,\hat{}) = (q_2,\hat{},-1)$

$\delta(q_2,\hat{}) = (q_5,\hat{},0)$
$\delta(q_2,a) = (q_3,\hat{},-1)$
$\delta(q_2,b) = (q_4,\hat{},-1)$

$\delta(q_3,a) = (q_3,a,-1)$
$\delta(q_3,b) = (q_4,a,-1)$
$\delta(q_3,\hat{}) = (q_5,a,0)$

$\delta(q_4,a) = (q_3,b,-1)$
$\delta(q_4,b) = (q_4,b,-1)$
$\delta(q_4,\hat{}) = (q_5,b,0)$

code(Tail) = 5,5,(0,a,1,blank,1),(0,b,1,blank,1),
(1,a,1,a,1),(1,b,1,b,1),(1,blank,2,blank,-1)
(2,blank,5,blank,0),(2,a,3,blank,-1), (2,b,4,blank,-1)
(3,a,3,a,-1),(3,b,4,a,-1),(3,blank,5,a,0)
(4,a,3,b,-1),(4,b,4,b,-1),(4,blank,5,b,0)

# Summary..Universal Turing Machines

Find a TM U such that for any TM M and input w to M

**$f_U$ (description of M * w) = $f_M$ (w).**

U needs to be able to read its input so we must standardise:

M is standard if

input alphabet = C
full alphabet = C $\cup$ {^}
it has 1 tape, 1-way infinite

so M has 1 track only with no implicit marking of square 0