Reduction or..

new unsolvable problems from old ones

e.g. does M halt on input □?

- the Empty Input Halting Problem (EIHP)

we show that any solution to the EIHP can be used to solve HP

..BUT **we know there is no solution to HP**

□ **there can be no algorithmic solution to EIHP.**

Reduction:

We say that problem A **reduces** to another problem, B,
if we can convert any Turing Machine solution to B into a Turing machine solution to A
…if we can show how to adapt a solution to B to give a solution to A.

We might say that solving A is no harder than solving B

So if we know there is no TM solution to A, we deduce that there can be no TM solution to B either.

..important to get this argument the right way round..

Unsolvable problems

by Church's Thesis:

unsolvable by a Turing Machine

⇒ there is no algorithmic solution

This is independent of future hardware developments (eg faster machines)

Proof Methods

1. assume a solution exists..derive a contradiction..deduce the assumption was wrong

2. by reduction of a problem known to be unsolvable.

The Turing Machine M[w]

(useful in unsolvability proofs)

M is a TM, w a word of C

M[w] is a new TM which..

1. overwrites its input with w
2. returns to square 0
3. runs M

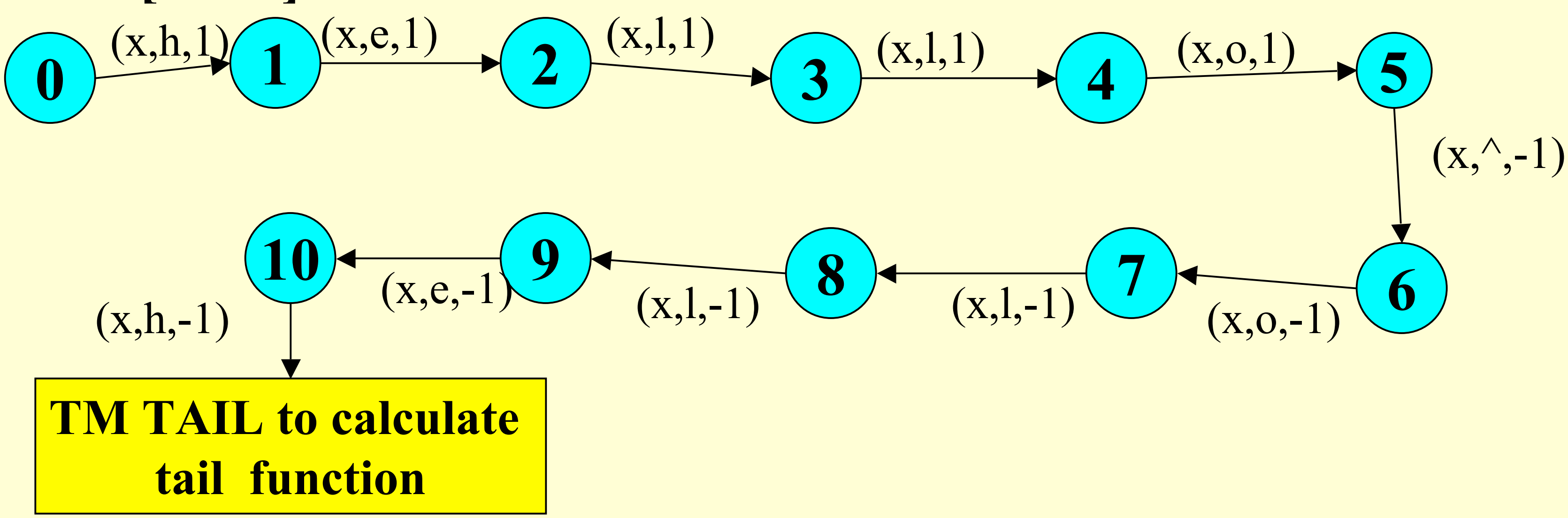
the input to M[w] is not relevant

- $f_{\mathbf{M[w]}}(v) = f_{\mathbf{M}}(w)$, any word v of I.
- M H & S on input w iff M[w] H & S on input v for any word v of I

If M is standard then M[w] can be made standard by
e.g. hard-wired return to square 0
(so no need to mark square 0 (non-standard).
e.g. for the Tail Turing Machine $f_{Tail}(w) = tail(w)$, all $w \in C^*$
(deletes first symbol of w & moves the rest 1 square left)

TAIL[hello] outputs ello on any input

Tail[hello]



The Turing Machine EDIT

EDIT generates code(S[w]) given code(S)*w):

For standard TM S, word w of C.
..by adding new 5-tuples to code(S) to write w, and return to square 0

this involves:

- new initial state
- 1+2.length(w) new states (=N, say)
- adding N to all state numbers and
- updating all 5-tuples (add N to all state numbers)

..ie. the TM EDIT edits code(S) to give code(S[w])

$\mathbf{f_{EDIT}(code(M)*w) = code(M[w])}$

EIHP - the Empty Input Halting Problem

Does a TM Halt & Succeed on empty input □?

i.e. is there a TM EI such that for any standard TM S:

f_{EI}(code(S)) = 1 if S Halts & Succeeds on input □

0 otherwise ?

This is proved by reducing HP to EIHP..

..we show that a solution to EIHP would provide a solution to HP.. known to be impossible..

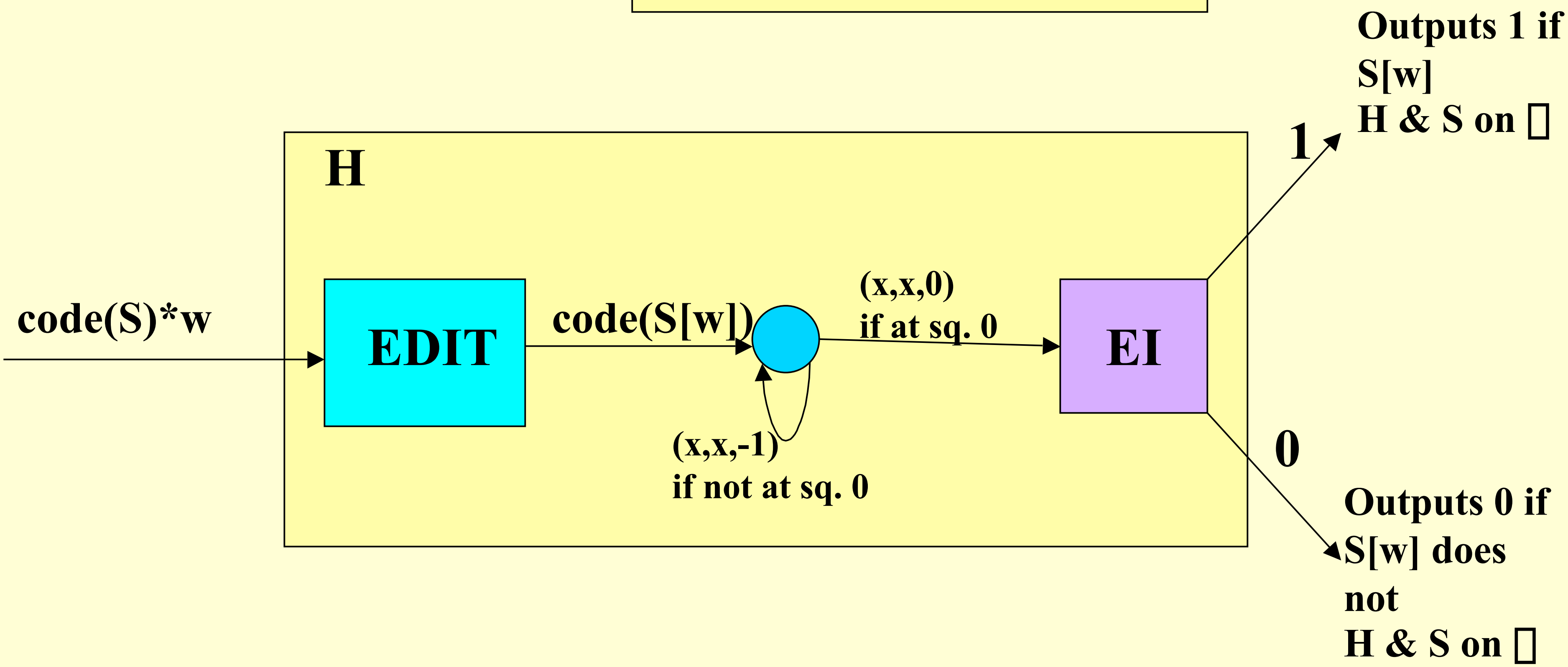
EI cannot exist.

EIHP is unsolvable

1. Assume TM EI to solve EIHP

2. Define TM H by…

run EDIT
return to square 0
run EI



Let H have input code(S)*w.
H runs EDIT□ code(S[w])..

…which is input to EI which outputs
1 if S[w] Halts & Succeeds on input □.
0 otherwise

S[w] Halts & Succeeds on input □
iff S Halts & Succeeds on input w

so: **H produces 1 on tape if S Halts & Succeeds on input w**
0 otherwise.

this is HP which has **NO** solution.
□ assumption that □EI is false □ EIHP is unsolvable.

HP has been reduced to EIHP, proving EIHP unsolvable.

Summary..unsolvability results:

- we proved the Halting Problem unsolvable directly
  - by **assuming a solution** and **deriving a contradiction** from this assumption

- we proved EIHP (empty input halting problem) unsolvable
  - by **reducing the Halting Problem to EIHP**
  - showing that any solution to EIHP would provide a solution to the Halting Problem..

    …previous unsolvability result.

These methods can be used to prove many related results