COSC1107 Computing Theory

(We will commence soon. We are just allowing a few minutes for people to join and set up. *Please mute your microphone unless you are speaking.* You can raise your hand or use the chat at any time.)

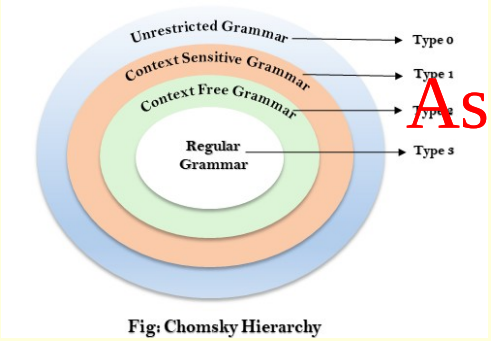
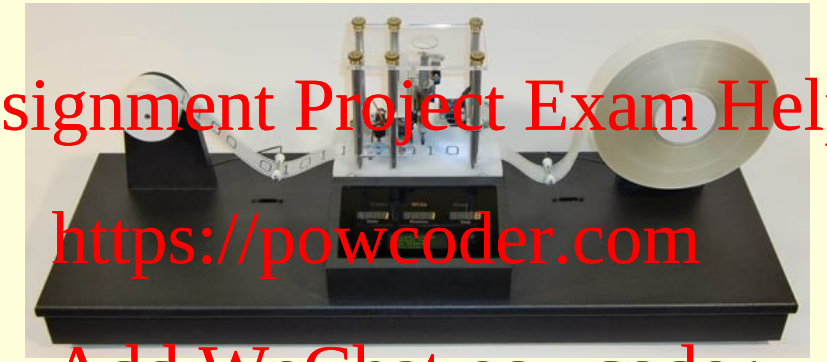
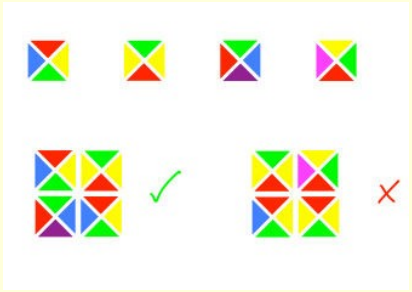
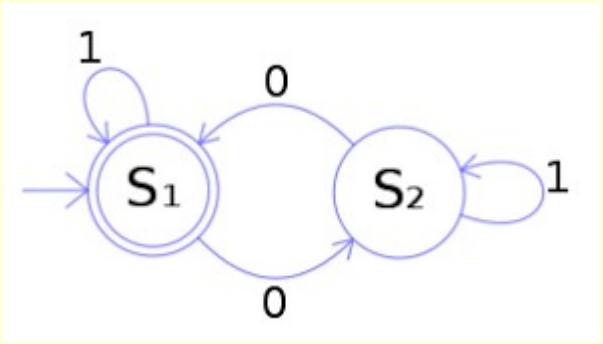
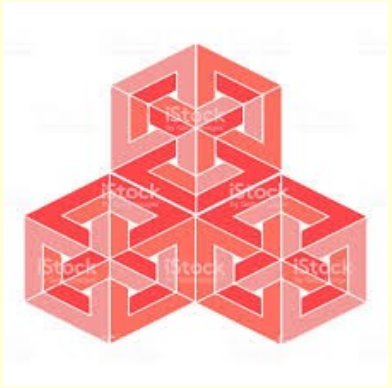


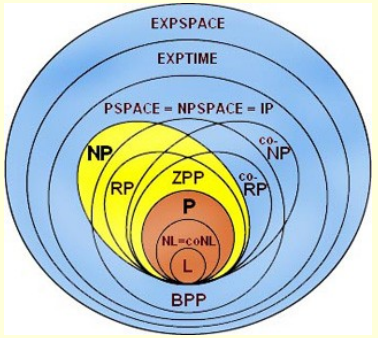
Fig: Chomsky Hierarchy

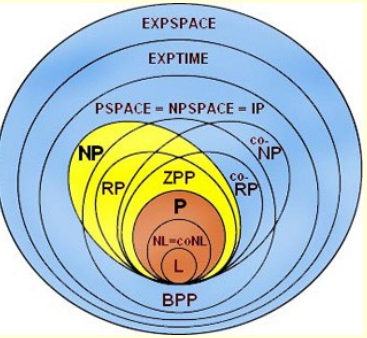


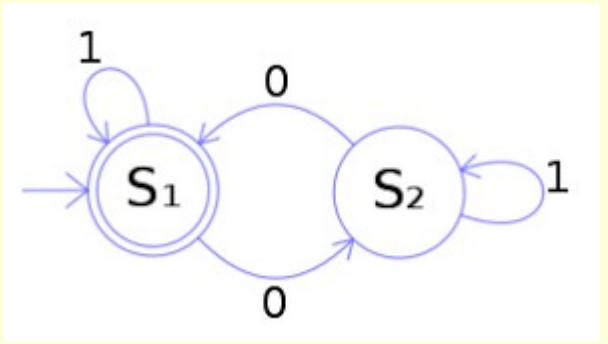












COSC1107

Computing Theory

Computability, Universality

Week 5

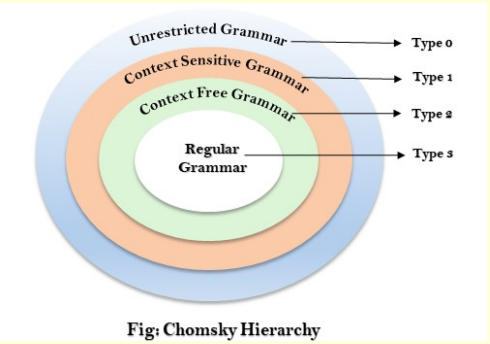


Fig: Chomsky Hierarchy

James Harland

james.harland@rmit.edu.au

* With thanks to Sebastian Sardina

Intro music 'Far Over' playing now …

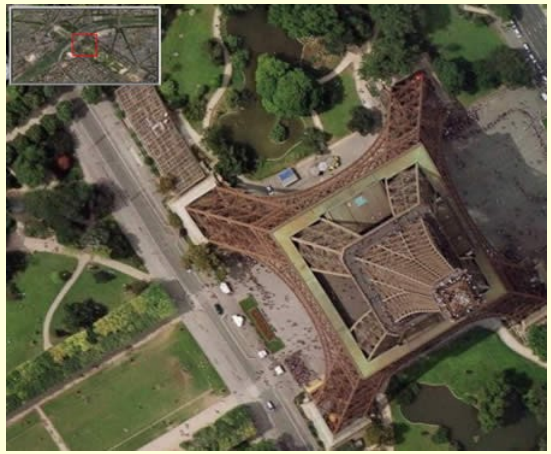
Acknowledgement

RMIT University acknowledges the people of the Woi wurrung and Boon wurrung language groups of the eastern Kulin Nations on whose unceded lands we conduct the business of the University. RMIT University respectfully acknowledges their Ancestors and Elders, past and present.

RMIT also acknowledges the Traditional Custodians and their Ancestors of the lands and waters across Australia where we conduct our business.

(add your name here to volunteer for this or email me)

Overview



- Questions?
- Universal Turing Machines
- Questions?
- Computability
- Questions?
- Platypus Game
- Questions?

What **can** be done

What **can't** be done

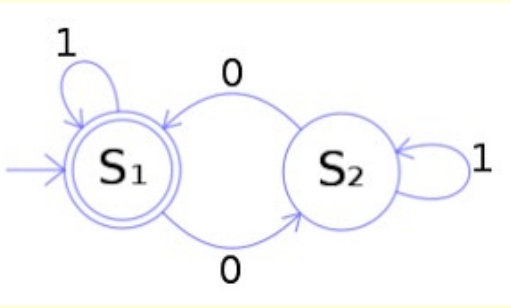
Both!

Questions?

Questions?

Questions?

Church-Turing thesis

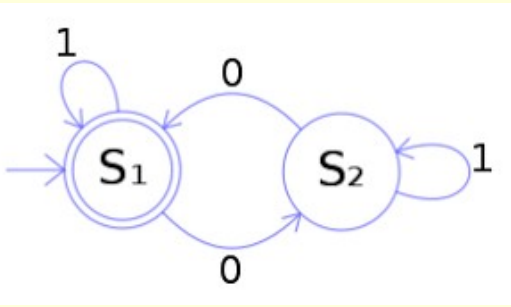


Church-Turing thesis (1936) *Any* computation can be translated to an equivalent computation on a Turing machine

so Turing machines are adequate for *anything* that can be computed

- No need to seek anything more powerful (there is nothing!)
- Enables rigorous analysis of computation
- Can't be done by a TM means can't be done at all!
- Thesis, not a theorem
- Observed property of the universe, like scientific laws
- Consistent with observation, no counterexamples known

Church-Turing thesis



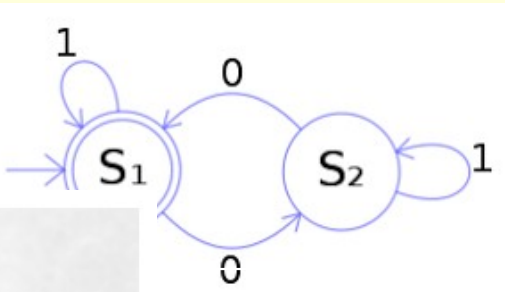
"A man* provided with paper, pencil, and rubber, and subject to strict discipline, is in effect a universal machine."
-- Alan Turing

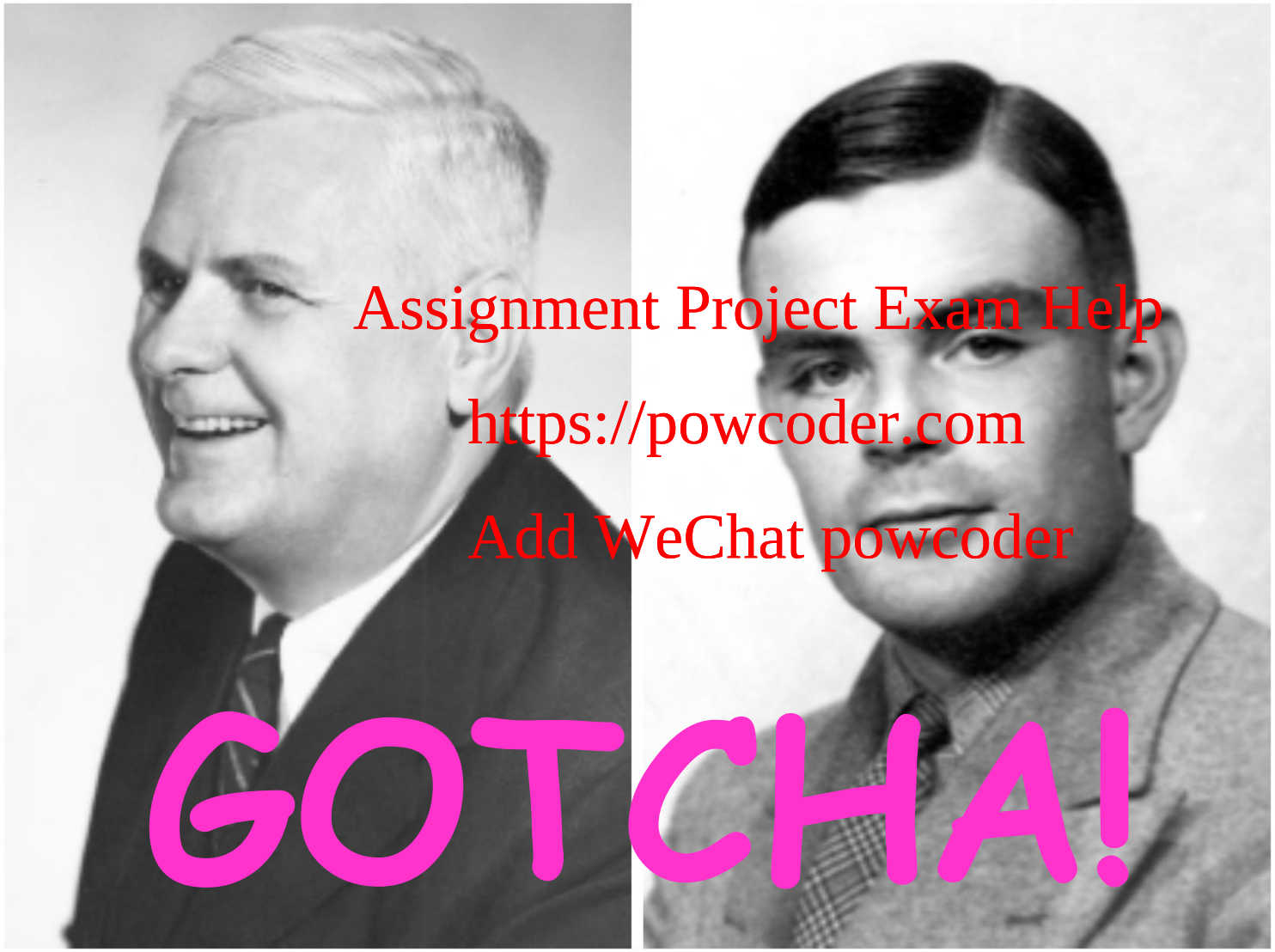
"The idea behind digital computers may be explained by saying that these machines are intended to carry out any operations which could be done by a human computer."
-- Alan Turing

"It is possible to invent a single machine which can be used to compute any computable sequence. If this machine U is supplied with a tape on the beginning of which is written the [encoding] of some computing machine M ,then U will compute the same sequence as M."
-- Alan Turing

* This now sounds very dated. But that is what he said back in the 1930's …

Church-Turing thesis





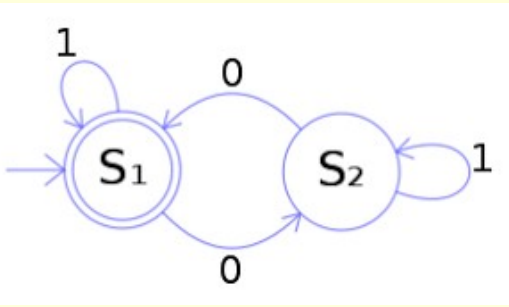
GOTCHA!

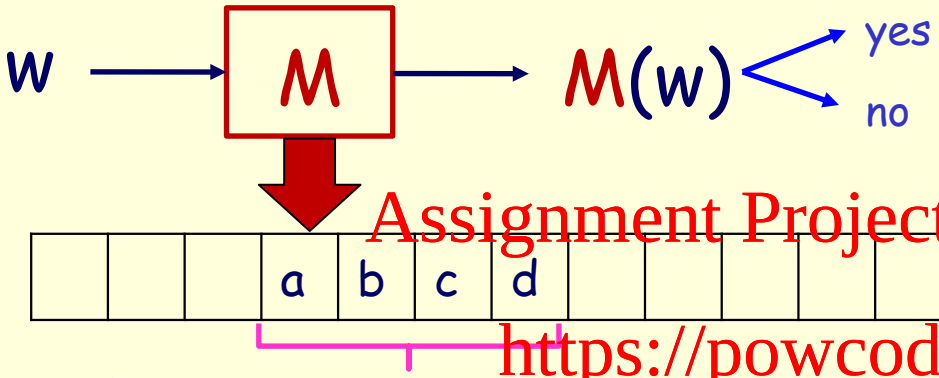
ed by linguists

l interactions

m states

Universal Turing machine

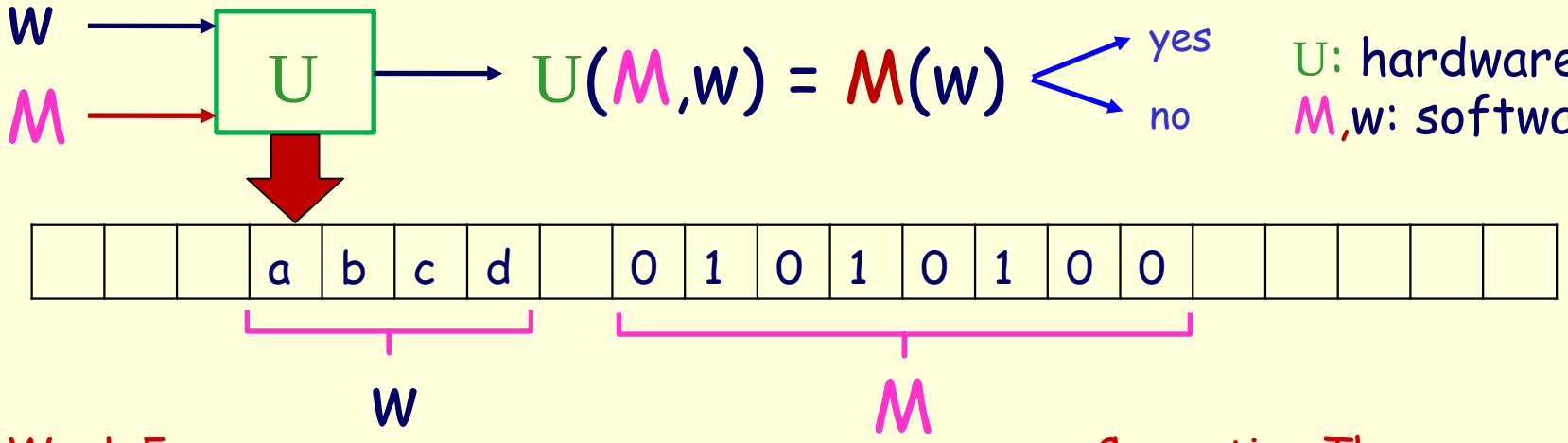




Solves one specific problem
New M for each new problem

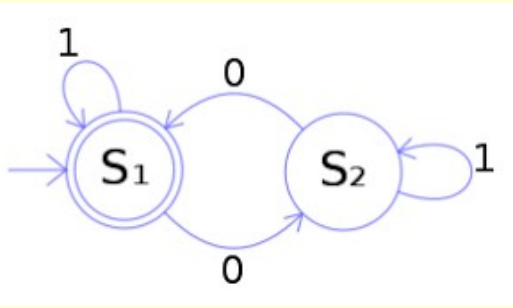
"U simulates the action of M on w"

Architecture that can solve any (computable) problem!



U: hardware
M,w: software

Universal Turing machine



I'd like a Turing machine please!

Certainly Master Baggins! Which one? We have many!

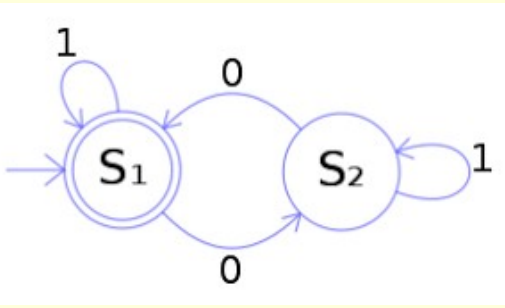
There are so many! It is confusing …

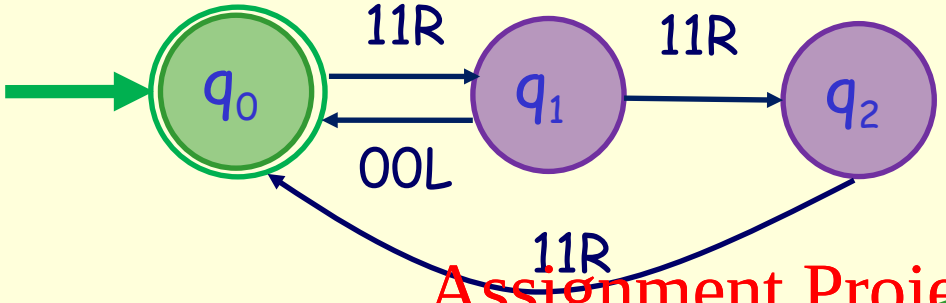
Frodo! Just use this one!
You will never need another!

U

"lean green Turing machine!"

Universal Turing machine





separate items by 0

T = (q,x,y,D,r)

en(T) = en(q)0en(x)0en(y)0en(D)0en(r)

q_0	1	1	R	q_1	10110110110 11
q_1	1	1	R	q_2	1101101101101 11
q_1	0	0	L	q_0	1101010101
q_2	1	1	R	q_0	1110110110110 1

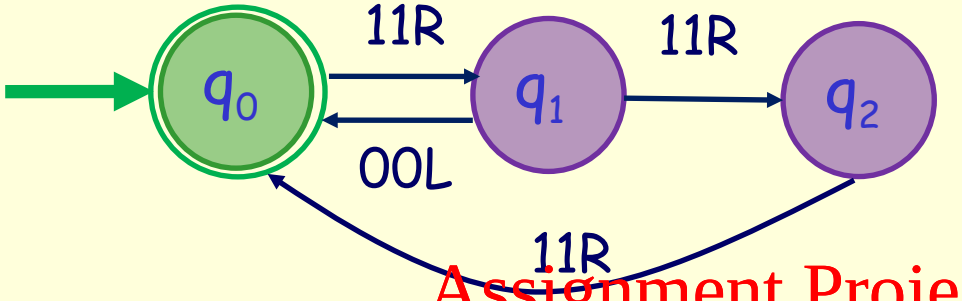
i	en(i)
q_0	1
q_1	11
q_2	111
0	1
1	11
	111
L	1
R	11

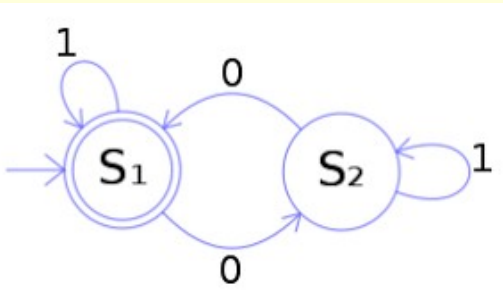
separate transitions by 00

code(M) = 000 en(T_1) 00 en(T_2) 00 en(T_3) 00 en(T_4) 000

start and end encoding with 000

Universal Turing machine





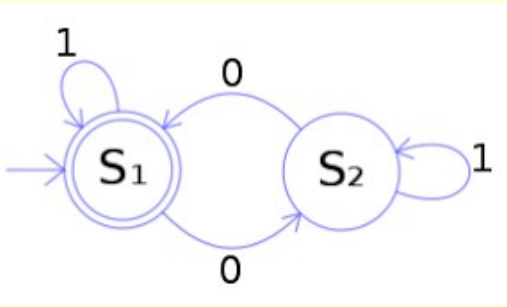
q_0	1	1	R	q_1	1011011011011
q_1	1	1	R	q_2	1101101101101 11
q_1	0	0	L	q_0	1101010101
q_2	1	1	R	q_0	1110110110110 1

i	en(i)
q_0	1
q_1	11
q_2	111
0	1
1	11
	111
L	1
R	11

code(M) = 000 en(T_1) 00 en(T_2) 00 en(T_3) 00 en(T_4) 000

000101101101101100110110110110111001101010101001110110110110100 0

Universal Turing machine

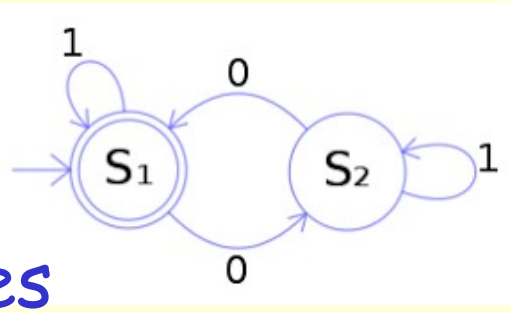


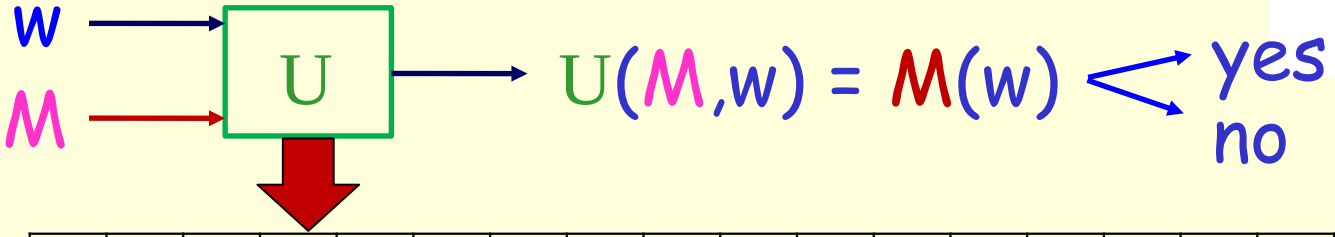
Turing machine string input to another TM

"Analyser" TMs can

- determine if (another) TM
  - has a 'halt' transition (ie no transition for a given pair of state & symbol)
  - is deterministic or not (or semi-deterministic)
  - contains a 'platypus' state
  - simulates an FSA (all moves R, no changing the tape)
  - ...
- Count the transitions, states, symbols, ... in a TM
- Add a transition to a TM
- Change a transition in a TM
- "Join" two TMs (final state of one = initial state of s
- Simulate the action of the input TM
- 'Swap' the acceptance behaviour of the input TM
- Change the output of the input TM
- Be given their own definition as input (!!)

Universal Turing machine





w M

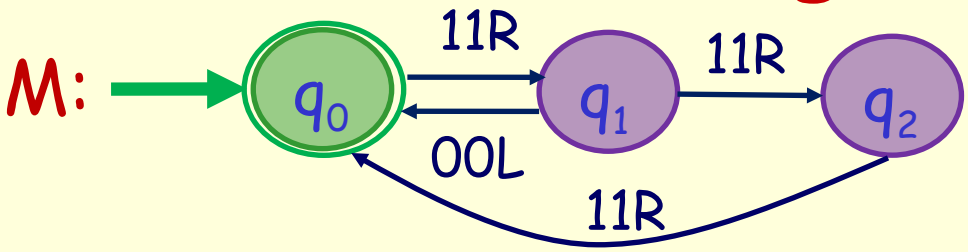
"One machine to rule them all;
one machine to find them;
one machine to …"

Build UTM using 3 tapes

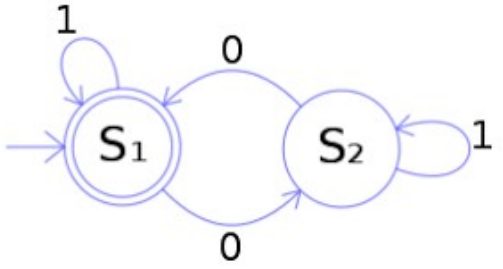
- One for input (encoded machine and input)
- One for the state of M
- One for the tape of M

Tape 1	code(M)code(w)
Tape 2	current state of M (encoded)
Tape 3	tape of M (encoded)

Universal Turing machine



w: 11001



i	en(i)
q_0	1
q_1	11
q_2	111
0	1
1	11
	111
L	1
R	11

M

U's alphabet may be different from M's ...

U:

code(M) code(w)

000101101101101100110110110110111001101010101001110110110110100011011010101100

U

code(M)code(w)

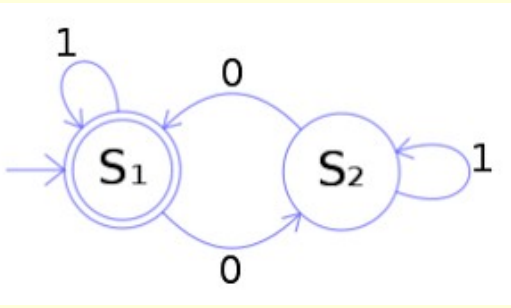
1

M state (encoded)

11011010101100

M tape (encoded)

Universal Turing machine

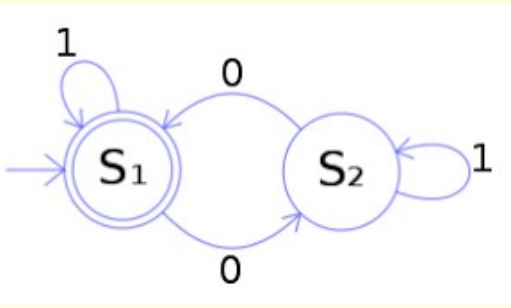


U:

1. Check tape 1. If format wrong, loop forever
2. Write code(w) on tape 3
3. Write en(q_0) on tape 1
4. Find transition matching q on tape 2 & x on tape 3. If none, halt.
5. Given en(q)0en(x)0en(y)0en(D)0en(r)
   - Replace en(q) with en(r) on tape 2
   - Write en(y) appropriate on tape 3
   - Move tape 3 according to D (left or right)
6. Go to step 4

000101101101101100110110110110111001101010101001110110110110100011011010101100

Universal Turing machine



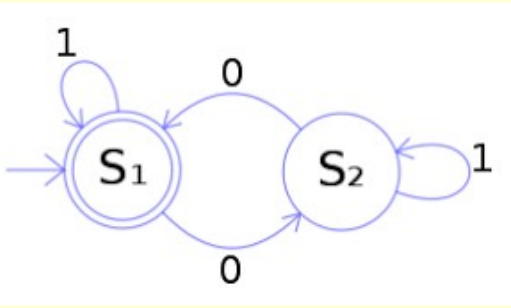
U:

1. Check tape 1. If format wrong, loop forever
2. **Write code(w) on tape 3**
3. Write en(q_0) on tape 1
4. Find transition matching q on tape 2 & x on tape 3. If none, halt.
5. Given en(q)0en(x)0en(y)0en(D)0en(r)
   - Replace en(q) with en(r) on tape 2
   - Write en(y) appropriate on tape 3
   - Move tape 3 according to D (left or right)
6. Go to step 4

000101101101101100110110110110111001101010101001110110110110100011011010101100

11011010101100

Universal Turing machine



U:

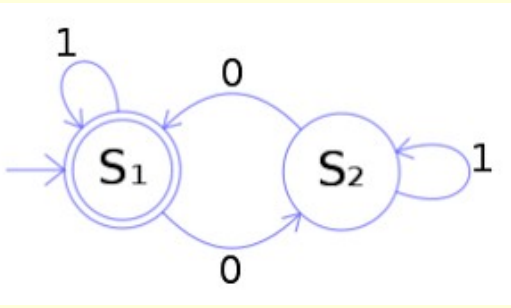
1. Check tape 1. If format wrong, loop forever
2. Write code(w) on tape 3
3. **Write en(q_0) on tape 1**
4. Find transition matching q on tape 2 & x on tape 3. If none, halt.
5. Given en(q)0en(x)0en(y)0en(D)0en(r)
   - Replace en(q) with en(r) on tape 2
   - Write en(y) appropriate on tape 3
   - Move tape 3 according to D (left or right)
6. Go to step 4

000101101101101100110110110110111001101010101001110110110110100011011010101100

1

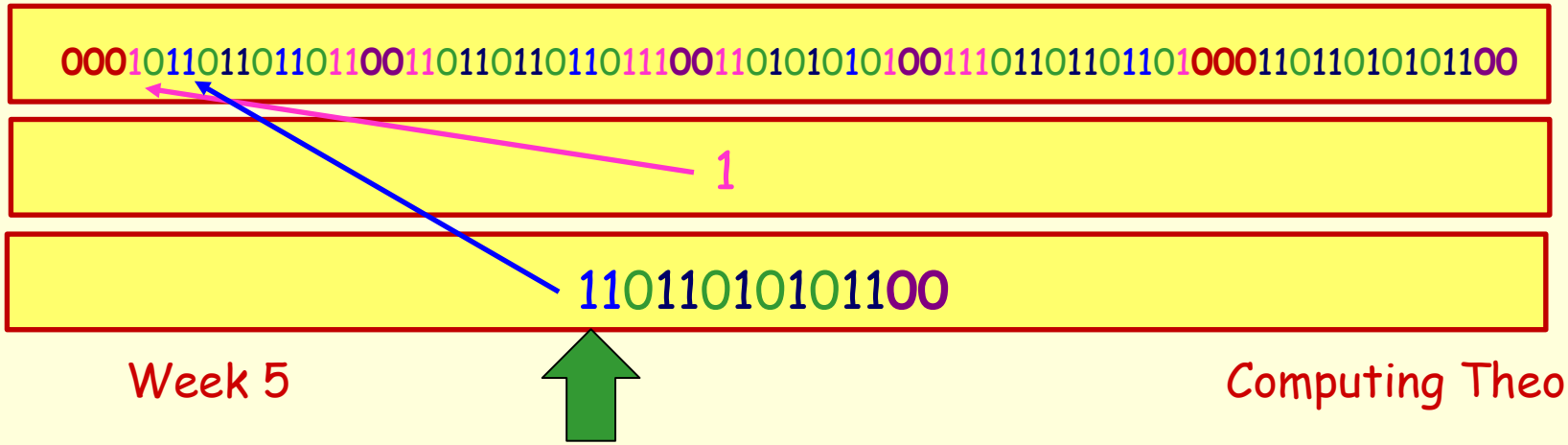
11011010101100

Universal Turing machine

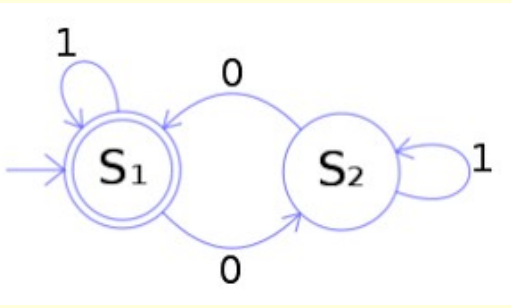


U:

1. Check tape 1. If format wrong, loop forever
2. Write code(w) on tape 3
3. Write en(q_0) on tape 1
4. Find transition matching q on tape 2 & x on tape 3. If none, halt.
5. Given en(q)0en(x)0en(y)0en(D)0en(r)
   - Replace en(q) with en(r) on tape 2
   - Write en(y) appropriate on tape 3
   - Move tape 3 according to D (left or right)
6. Go to step 4

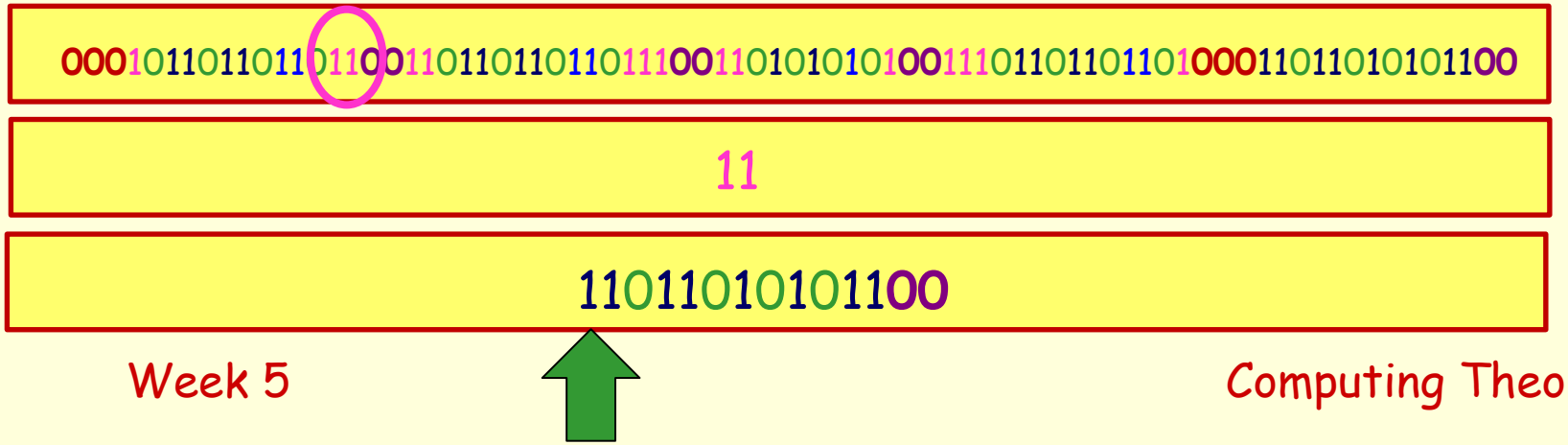


Universal Turing machine

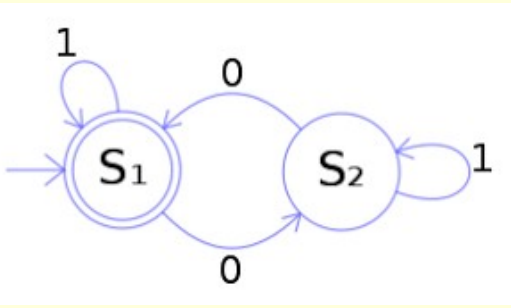


U:

1. Check tape 1. If format wrong, loop forever
2. Write code(w) on tape 3
3. Write en(q_0) on tape 1
4. Find transition matching q on tape 2 & x on tape 3. If none, halt.
5. Given en(q)0en(x)0en(y)0en(D)0en(r)
   - Replace en(q) with en(r) on tape 2
   - Write en(y) appropriate on tape 3
   - Move tape 3 according to D (left or right)
6. Go to step 4

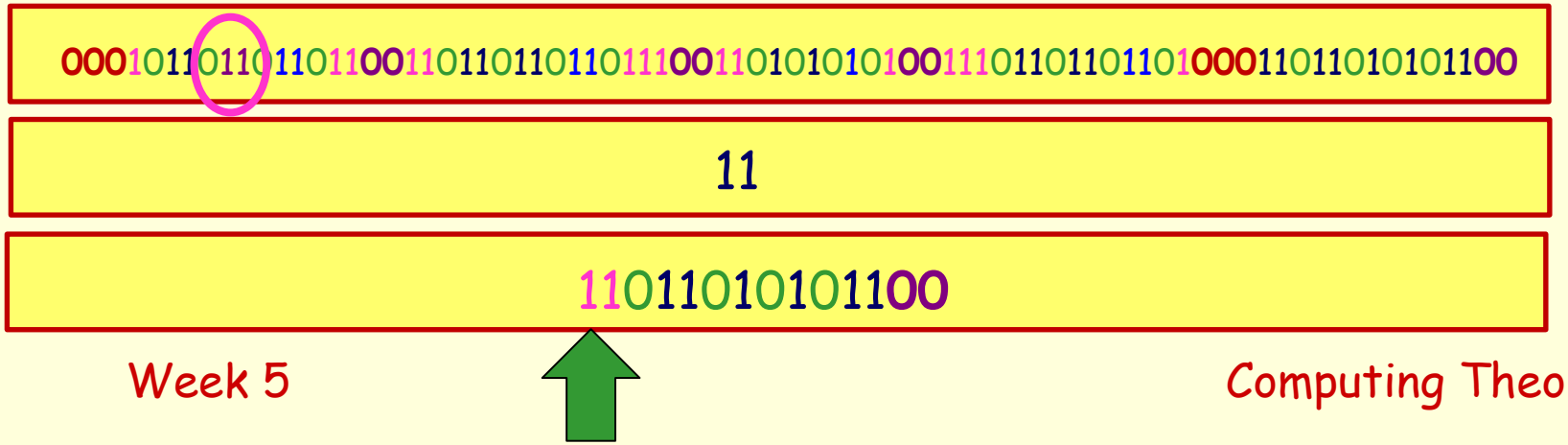


Universal Turing machine

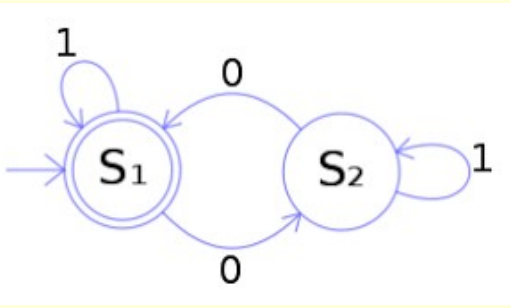


U:

1. Check tape 1. If format wrong, loop forever
2. Write code(w) on tape 3
3. Write en(q_0) on tape 1
4. Find transition matching q on tape 2 & x on tape 3. If none, halt.
5. Given en(q)0en(x)0en(y)0en(D)0en(r)
   - Replace en(q) with en(r) on tape 1
   - **Write en(y) appropriate on tape 3**
   - Move tape 3 according to D (left or right)
6. Go to step 4

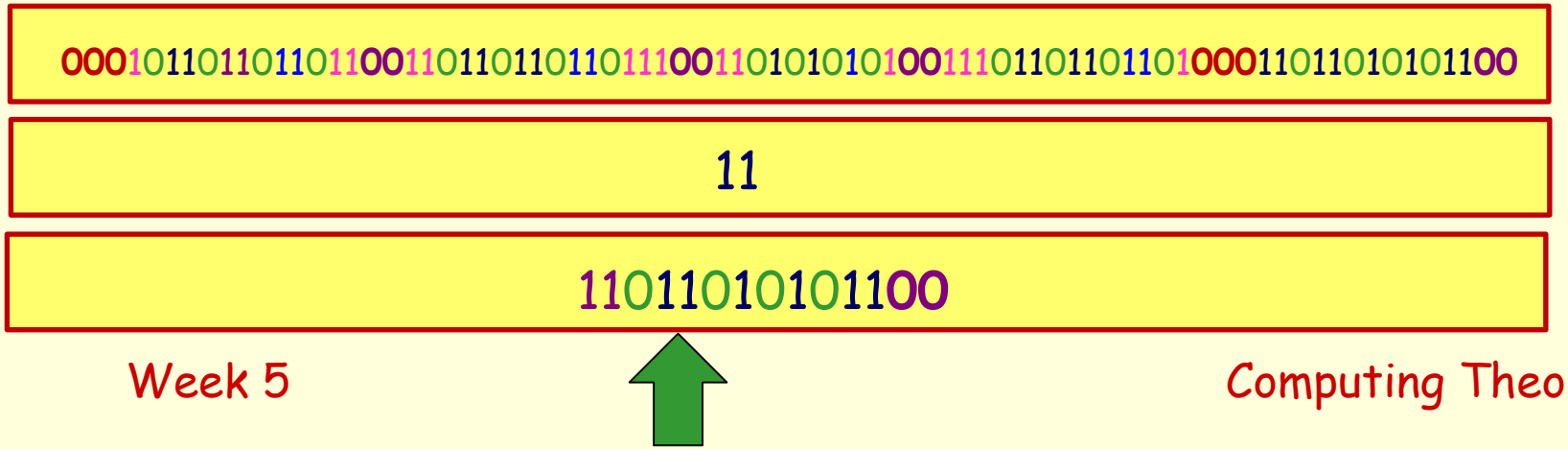


Universal Turing machine

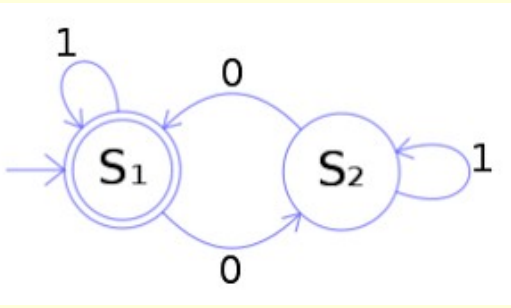


U:

1. Check tape 1. If format wrong, loop forever
2. Write code(w) on tape 3
3. Write en(q_0) on tape 1
4. Find transition matching q on tape 2 & x on tape 3. If none, halt.
5. Given en(q)0en(x)0en(y)0en(D)0en(r)
   - Replace en(q) with en(r) on tape 2
   - Write en(y) appropriate on tape 3
   - Move tape 3 according to D (left or right)
6. Go to step 4



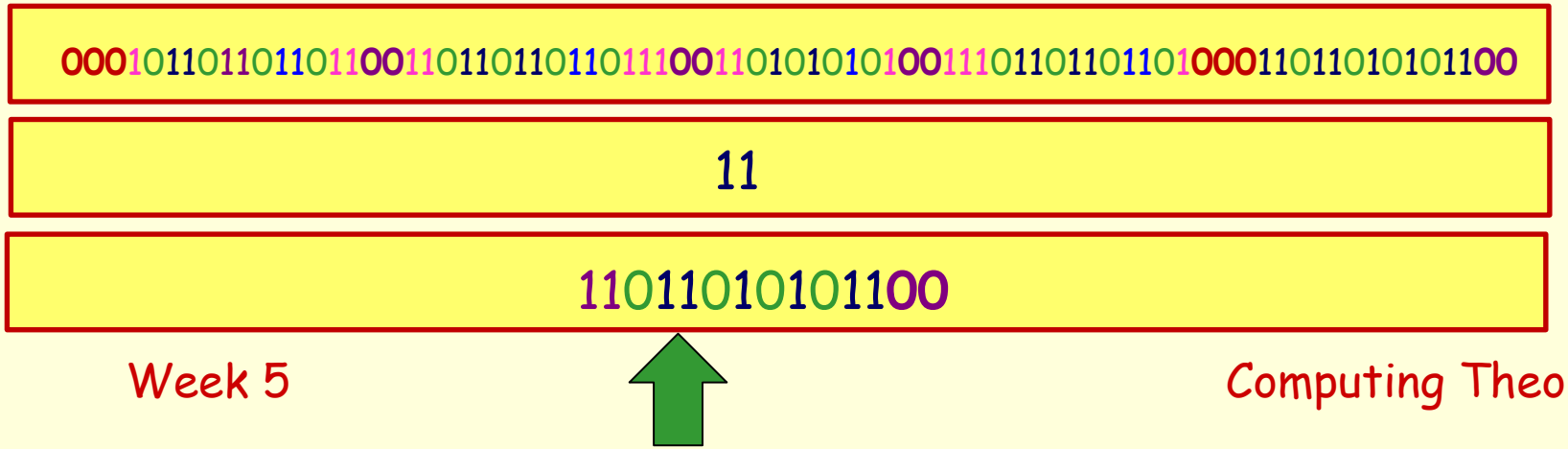
Universal Turing machine



U:

1. Check tape 1. If format wrong, loop forever
2. Write code(w) on tape 3
3. Write en(q_0) on tape 1
4. Find transition matching q on tape 2 & x on tape 3. If none, halt.
5. Given en(q)0en(x)0en(y)0en(D)0en(r)
   - Replace en(q) with en(r) on tape 2
   - Write en(y) appropriate on tape 3
   - Move tape 3 according to D (left or right)
6. Go to step 4

AND SO ON ...



Questions?

Questions?

Questions?

Impossible? Never?

Mary said Tom would never call her again, but I told her, "**Never say never**."
-- 'Pickwick Papers' by Charles Dickens

"Never is too long a word even for me..."
--Treebeard in 'The Lord of the Rings' by J.R.R. Tolkien

~~"Believe the unbelievable.~~
~~Dream the impossible.~~
~~Never take 'no' for an answer!"~~

If there is convincing evidence that something is impossible, ...

Week 5

Impossible? Never?

When can you ever say 'never!'?

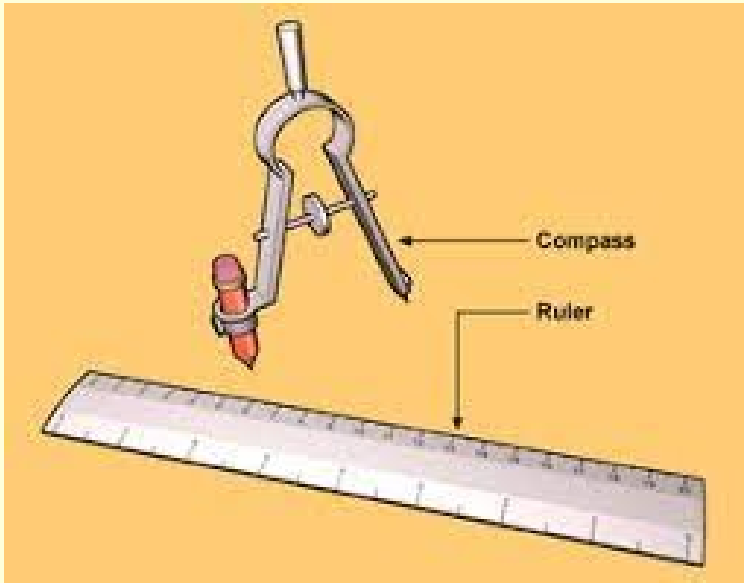
Solutions of polynomial equations

- 'closed' formulae exist for quadratic, cubic & quartic
- No such general formula can exist for degree ≥ 5 (!!)

Geometric construction of polygons and other shapes

- Only 'straightedge and compass' allowed

- Impossible constructions
  - 'Square the circle'
  - 'Double the cube'
  - Angle trisection



Impossible? Never?

When can you ever say 'never!'?

Clever people have got this wrong in the past!

Many predictions along the lines of 'this will never happen' have been incorrect...

So can you ever say 'never'?

Impossible? Never?

???

Known to be impossible

Riemann Hypothesis
P = NP
Computers pass Turing test
…

Status unknown

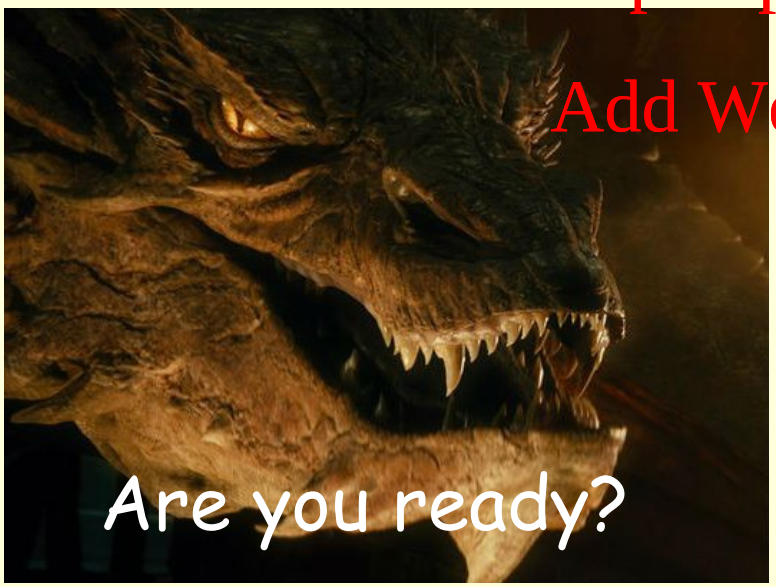
Computers outperform humans at chess
Self-driving cars
Text to speech and speech to text
Computer facial recognition
…

Known to be possible

Quiz time!

Go to Canvas and find the quiz Lectorial 5 Question set

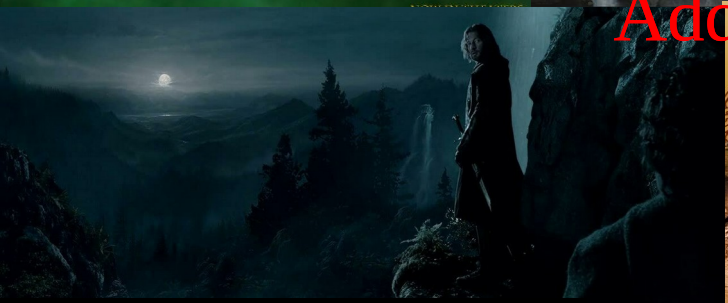
- Not worth any marks
- You can consult other students if you wish
- Time limit will be 10 minutes

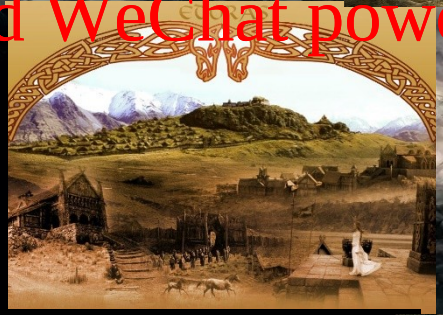


Go!

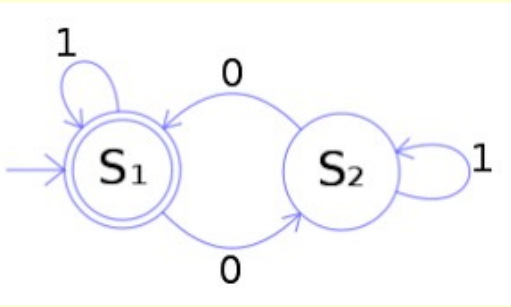
The pictures will take 10 minutes to disappear!

Thomas music means 1 minute left!





Decision problems



A decision problem is a **question** in a **formal system** with a **yes-no answer**, depending on some input parameters.

Sort: Is a list of numbers sorted?

Hamiltonian circuit: Is there a circular path in this graph that visits every node?

Primality: Is a given number prime?

Python syntax: Is a given program syntactically correct?

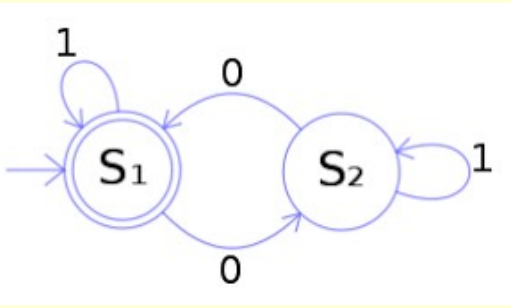
Password: Is the given password correct?

Bank balance: Is the transaction approved?

Halting: Given a TM M and input w, does M halt on w?

Harder than it may seem …

Decision problems

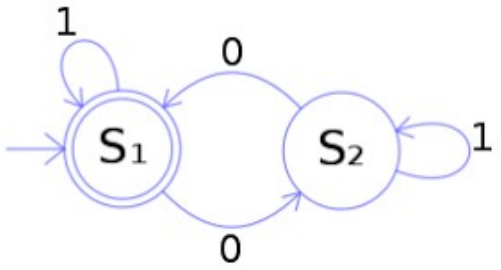


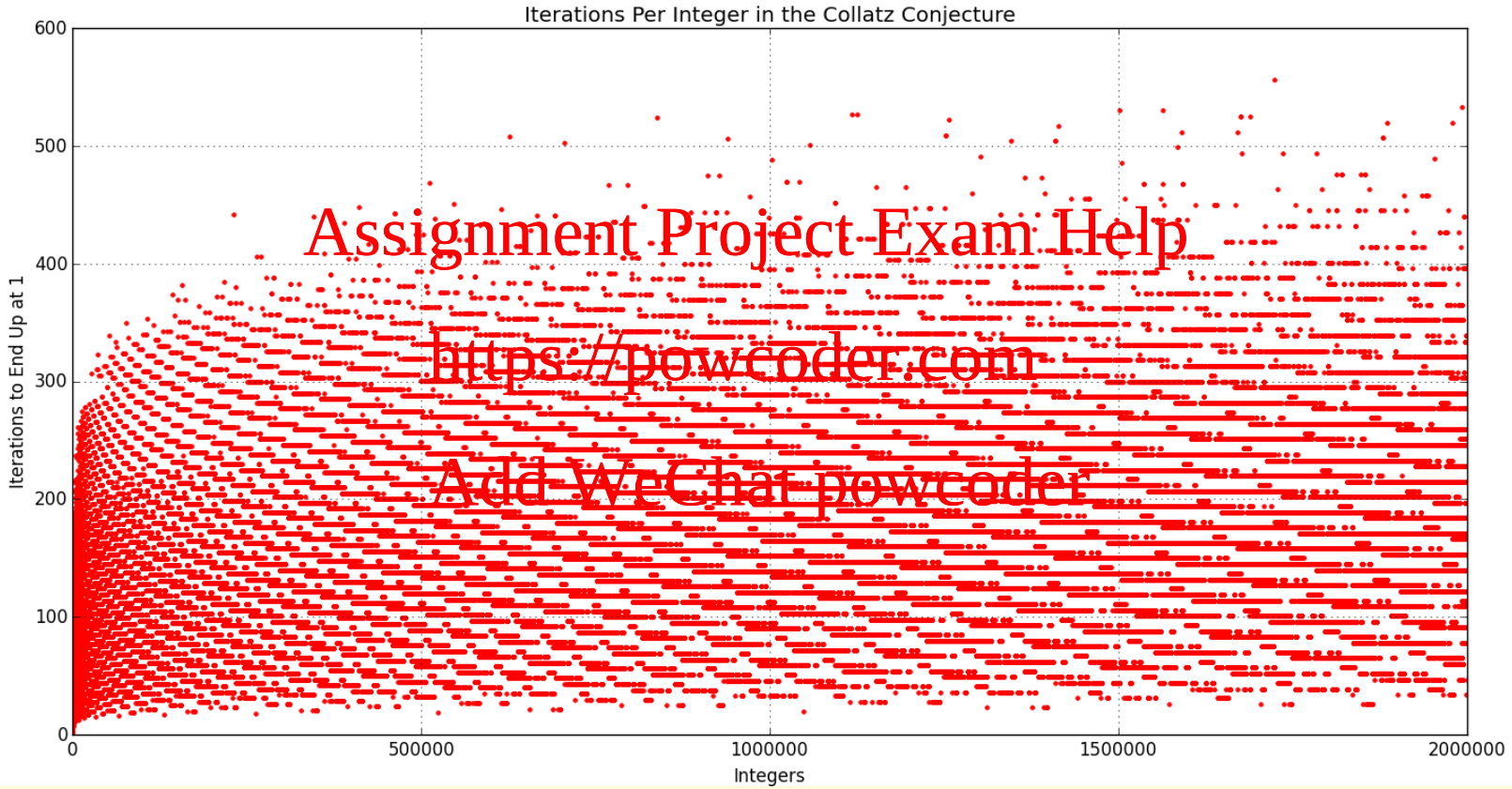

```
% n is an integer ≥ 1
while n > 1 do
        if n mod 2 == 0
                n = n/2
        else
                n = 3*n + 1
elihw
```


5 16 8 4 2 1
6 3 10 5 16 8 4 2 1
7 22 11 34 17 52 26 13 40 20 10 5 … 1
9 28 14 7 … 1
15 46 23 70 35 106 53 160 80 40 20 … 1

Does it always terminate? **No-one knows (!!)**

Decision problems

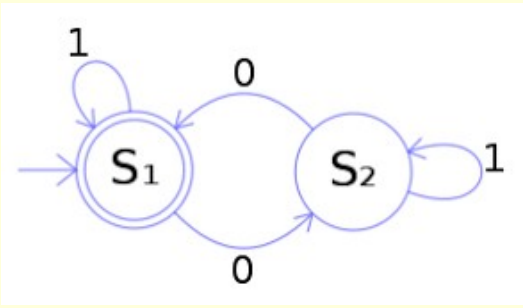




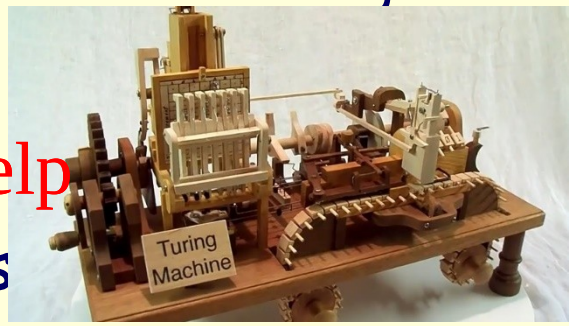
Does it always terminate?

No-one knows (!!)

Decidable problems

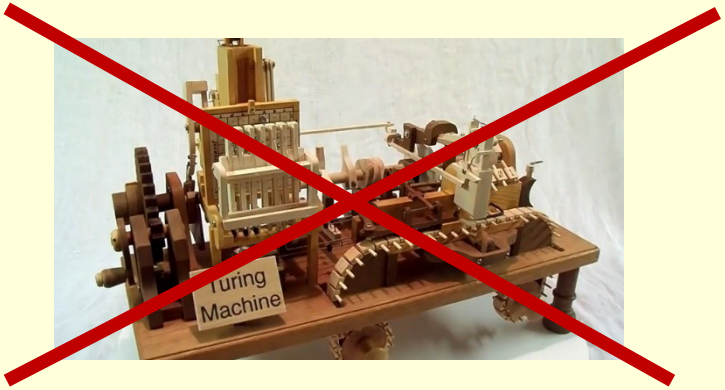


A decision problem is **decidable** if there **exists** a Turing machine M that solves it, ie

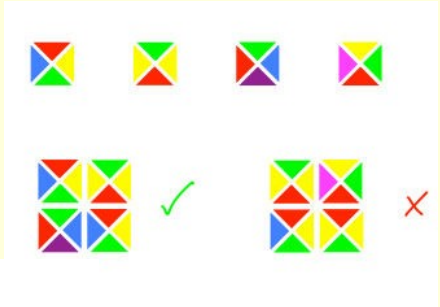


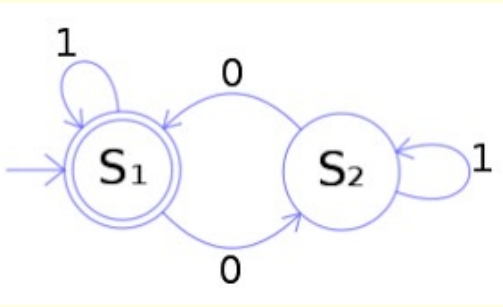
- M halts on all inputs w
- M outputs yes if the decision for w is yes
- M outputs no if the decision for w is no
- M produces no other outputs (so M is a language recogniser)

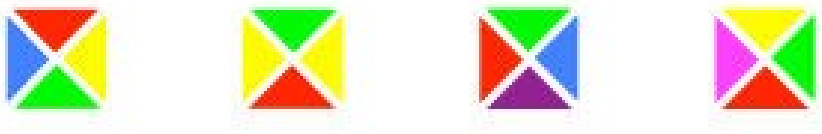
A decision problem is **undecidable** if there **does not exist** a Turing machine M that solves it.

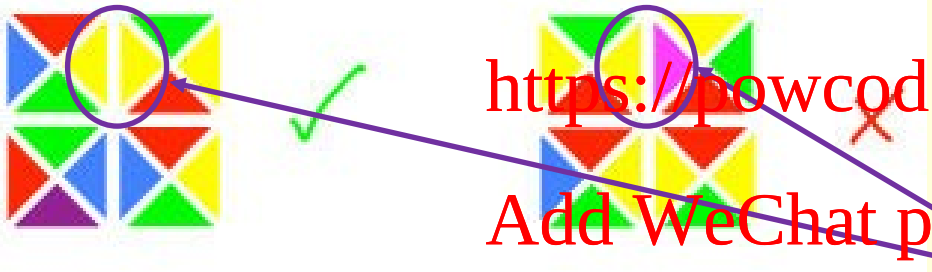


Tile problem







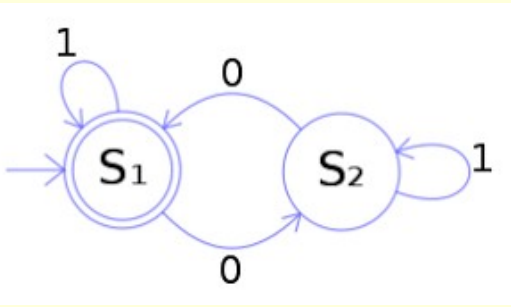


Can you cover any rectangular area with a given set of tiles?

- Tiles cannot be rotated
- Edge colours must match
- Tiles can be copied

There is no Turing machine that solves this decision problem (!!!)

Halting problem



Halting problem: Given a Turing machine M and an input w, does M halt on w?

For every TM M and every input w, either

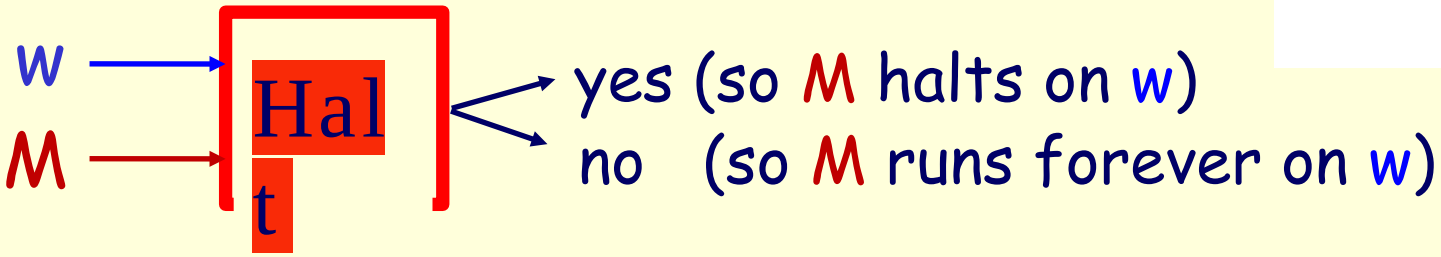
- M halts on w
- M doesn't halt on w

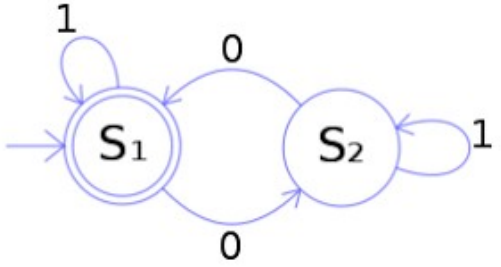
The Halting problem is a decision problem about Turing machines ..

Question: Is there a Turing machine that can solve the Halting problem for Turing machines?

Is the Halting problem decidable?

Halting problem





Is there a TM Halt such that ??

- Halt terminates on every input (ie every M and every w)
- Halt accepts code(M)code(w) if M terminates on w
- Halt rejects code(M)code(w) if M does not terminate on w

Option 1:

Halt exists

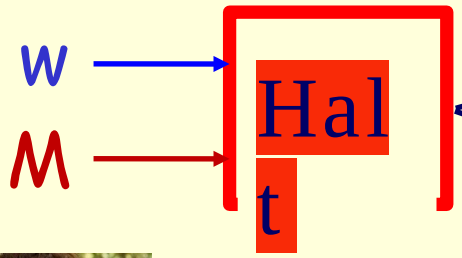
The Halting problem is decidable

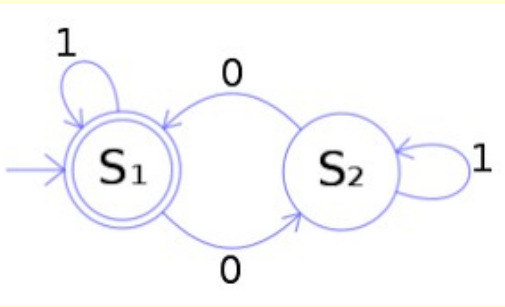
Option 2:

Halt does not exist

The Halting problem is undecidable

Halting problem





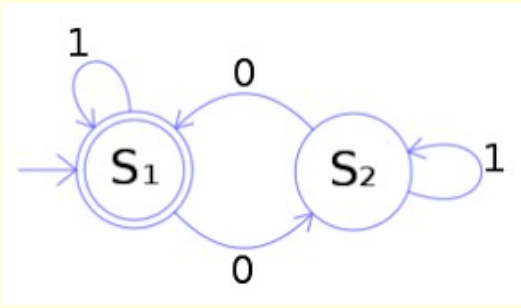
"Gandalf, does the machine Hal exist?"

"Let's ask my friend, Frodo …"

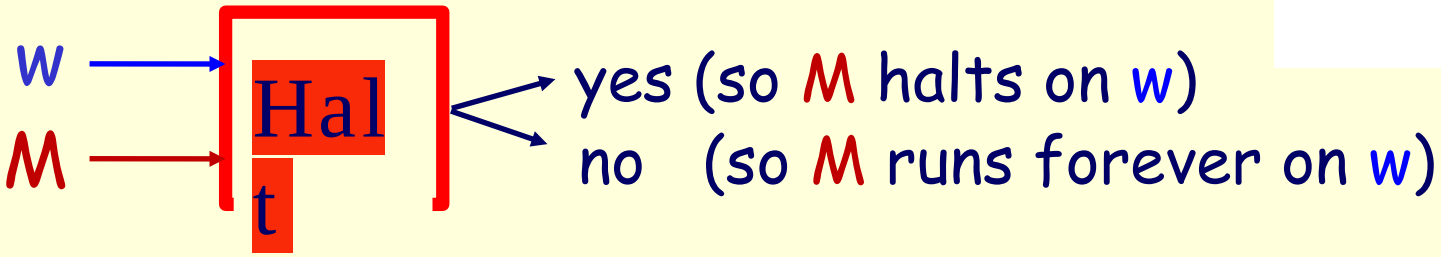
NO!!

Clues:

- "This statement is false" (Epiminedes)
- Paradox of the barber
- Self-reference



Halting problem



The Halting problem is undecidable

Proof: Assume that the TM Halt exists, ie

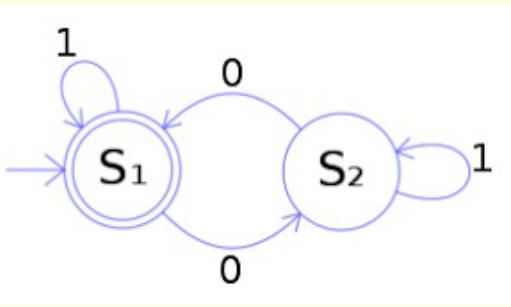
- Halt terminates on every input (ie every M and every w)
- Halt accepts code(M)code(w) if M terminates on w
- Halt rejects code(M)code(w) if M does not terminate on w

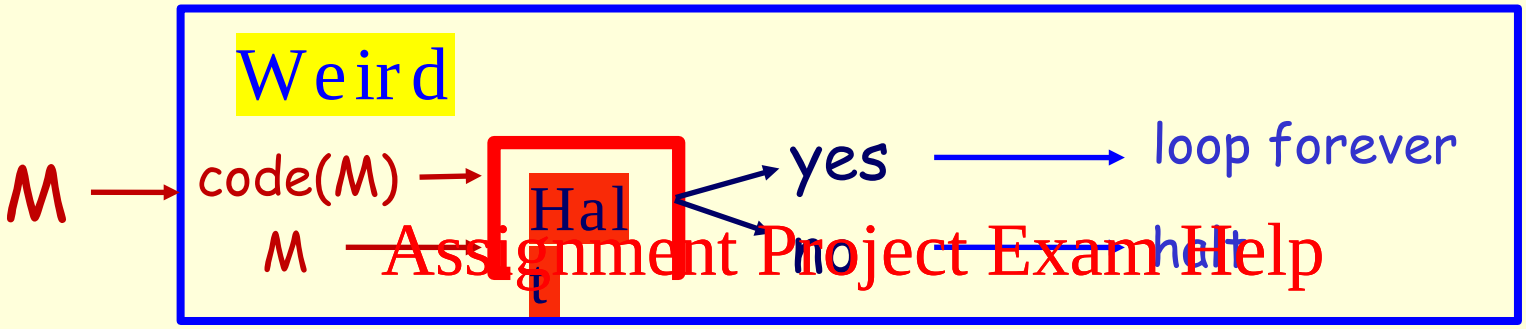
Use Halt to construct TM Weird as follows:

- If Halt accepts code(M)code(M)* then loop forever
- If Halt rejects code(M)code(M)* then halt

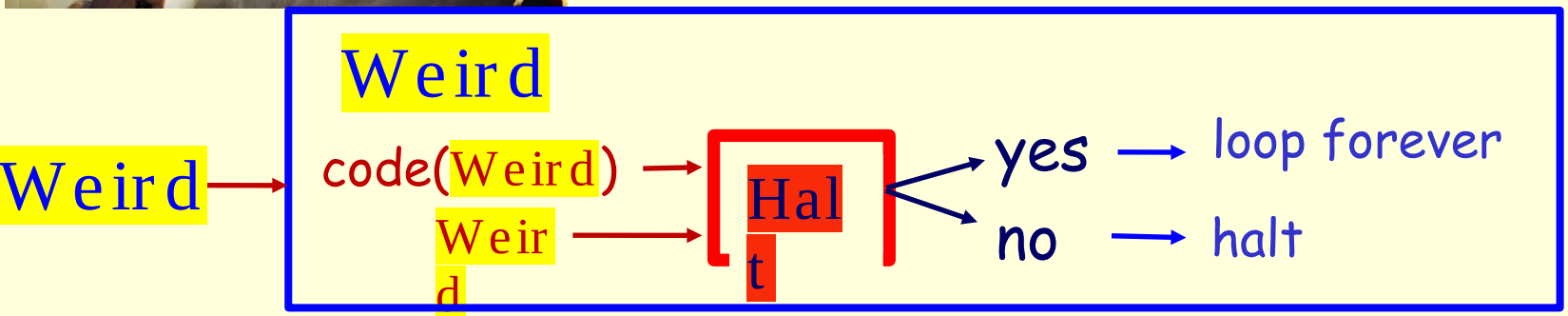
*encode M is as an input to itself

Halting problem

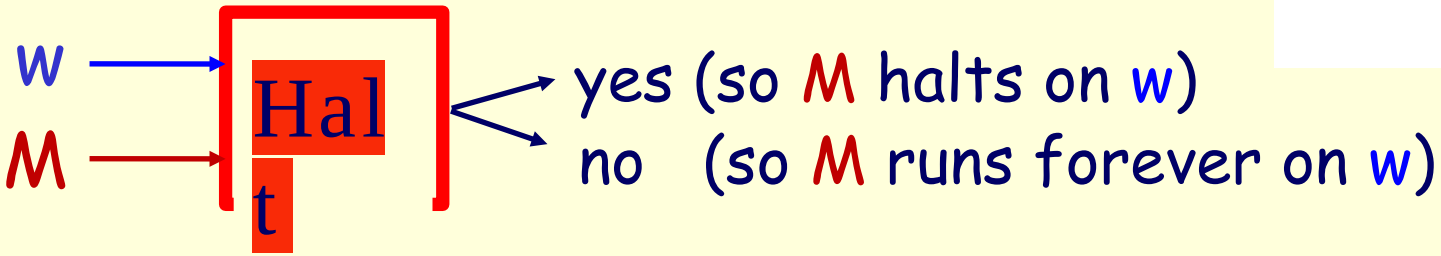


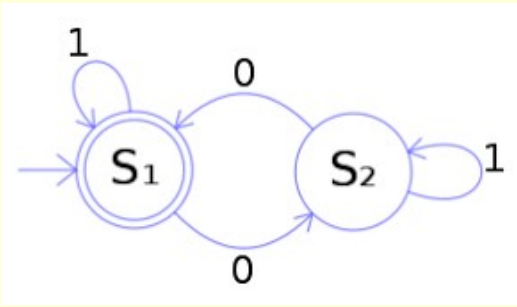


What if M = Weird ?



Halting problem





So if TM Halt exists, then TM Weird must also exist

Weird takes a TM M as input and

- If M halts on code(M), then Weird does not halt on code(M)
- If M does not halt on code(M), then Weird halts on code(M)

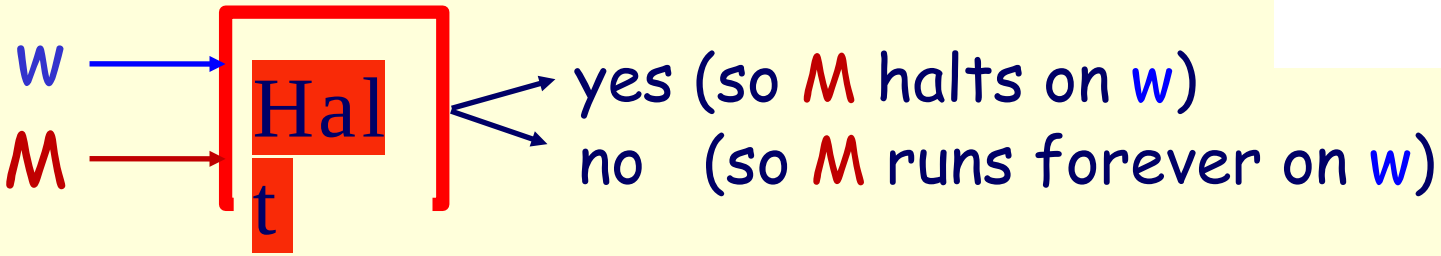
This must work for any TM M … Including when M = Weird (!!)

- If Weird halts on code(Weird), then Weird does not halt on code(Weird)
- If Weird does not halt on code(Weird), then Weird halts on code(Weird)

That's too weird!

CONTRADICTION!

Halting problem

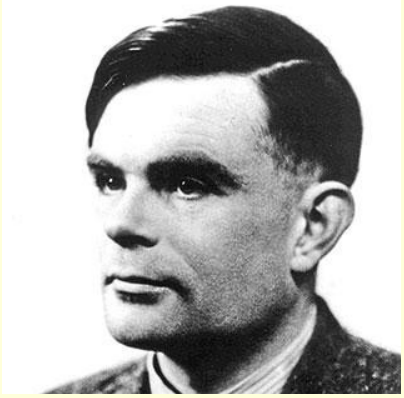


So our assumption is wrong, ie the TM Halt does not exist

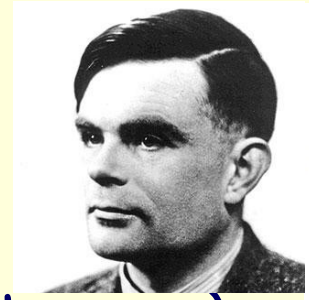
The Halting problem is therefore undecidable

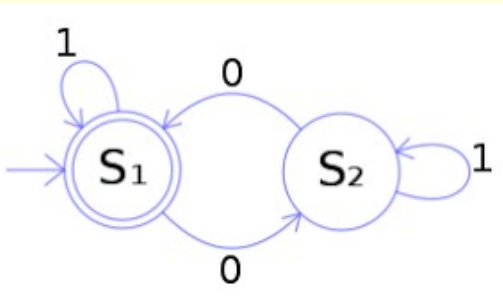
This means that there is no TM which

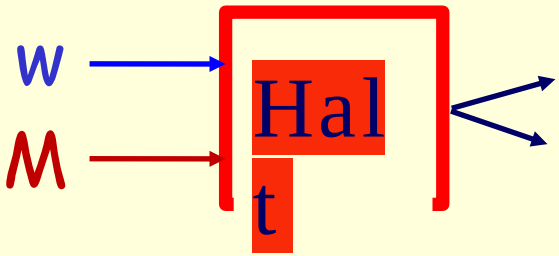
- Always halts for any M and w
- Outputs 'yes' if M halts on w
- Outputs 'no' if M does not halt on w
- (Does not produce any other output)



Halting problem







yes (so M halts on w)
no (so M runs forever on w)

This means any attempt to solve the halting program must for some M and w either:

1. Not halt
2. Halt and output no if M halts on w (!!)
3. Halt and output yes if M doesn't halt on w (!!)
4. Halt and produce some other output (e.g., "Dunno!")

1 is undesirable, 2 and 3 are insane!

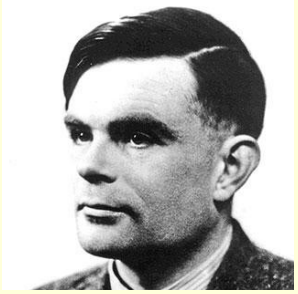
Essentially, we have to accept that an answer **"Don't know"** is sometimes unavoidable …

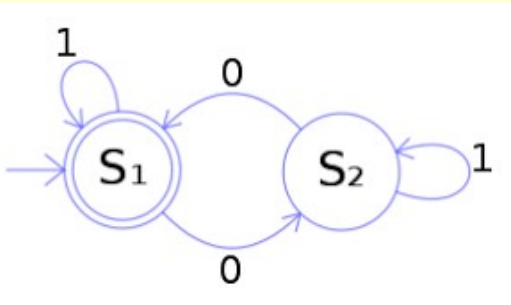
Questions?

Questions?

Questions?

Computability





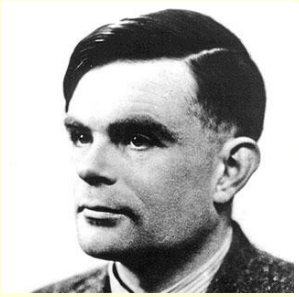
Decidable

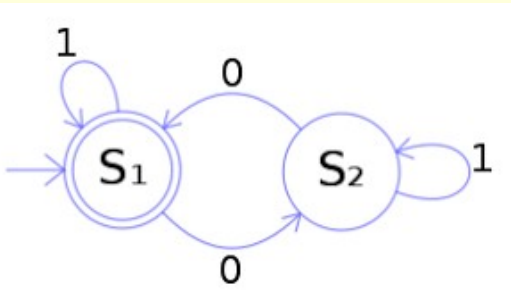
- Primality testing
- Factorisation
- Hamiltonian circuit
- 3-SAT satisfiability
- Sorting
- Travelling Salesperson
- …

Undecidable

- Does M halt on w? (halting problem)
- Does M halt on any w?
- Does M halt on all w?
- Does M halt on blank input?
- Do M_1 and M_2 halt on the same strings?
- Does M reach a particular state on input w?
- …
- Busy beaver problem
- Tile problem
- …

Computability





Let Problem *A* be a problem known to be undecidable, and Problem B with status unknown.

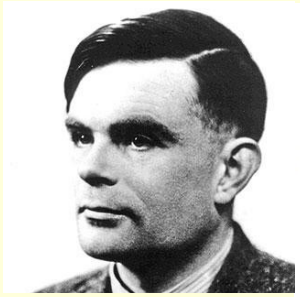
- Assume Problem B is decidable and derive a contradiction
- Reduce Problem *A* to Problem B

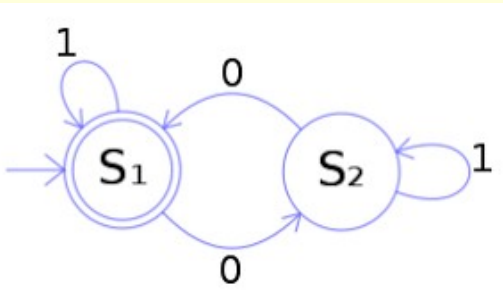
Problem reduction is a shortcut:

1. Show that a TM for Problem B can be used as part of a decision procedure for Problem *A*
2. Shows Problem B decidable Problem A decidable
3. Problem A undecidable
4. So Problem B undecidable

Step 1 is the only necessary part …

Computability



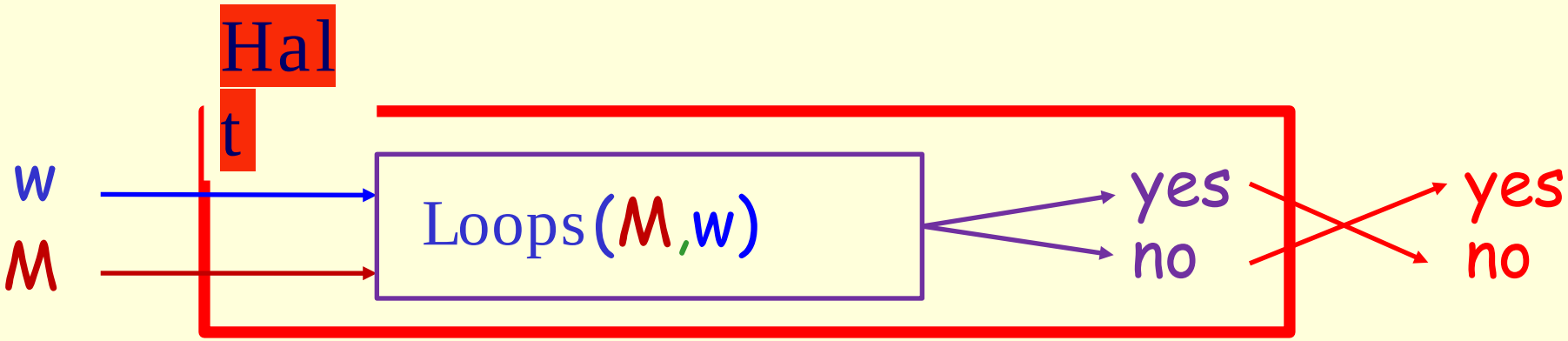


Loops problem: Does TM M run forever on input w?

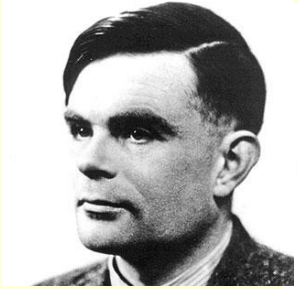
Assume there is a TM Loops which solves the Loops problem
Given M and w

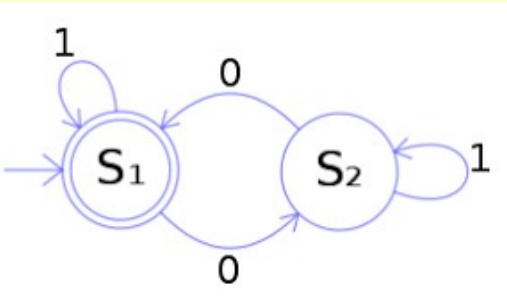
1. Run Loops on M and w
2. If Loops says yes then output no
3. If Loops says no then output yes

So the Loops problem is undecidable



Computability





Blank Tape problem: Does TM M halt on the blank tape?

Assume there is a TM Blank which solves the blank tape problem

Given M and w, compute machine N such that N with a blank tape

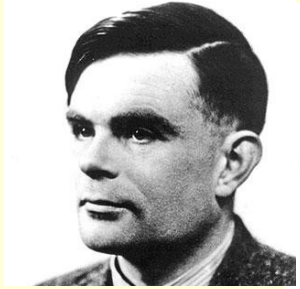
1. Writes w on the tape
2. Positions tape head to the first symbol in w
3. Goes to start state of M
4. Runs M on w

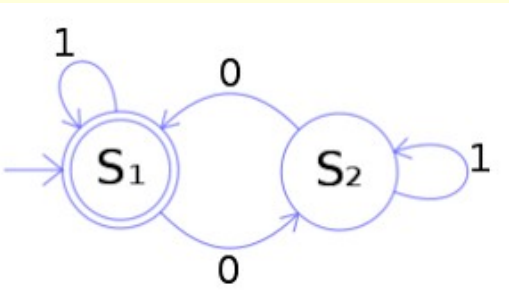
N halts on the blank tape iff M halts on w

So Blank on input N will solve the Halting Problem for M on w

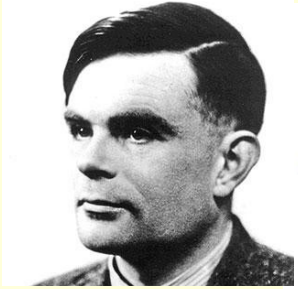
So the Blank Tape problem is undecidable

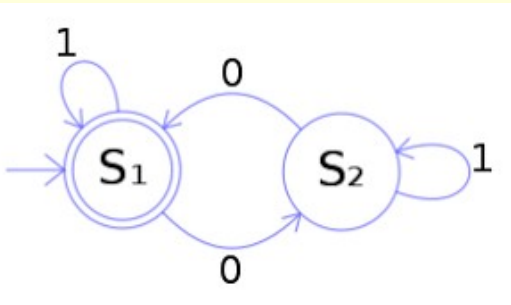
Computability





Computability





All Inputs problem: Does TM M halt on all inputs?

Assume there is a TM All which solves the All Inputs problem

Given M and w, compute machine O such that O

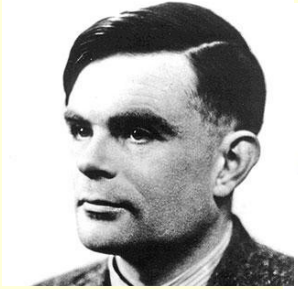
1. Erases its input
2. Writes w on the tape
3. Positions tape head to the first symbol in w
4. Goes to start state of M
5. Runs M on w

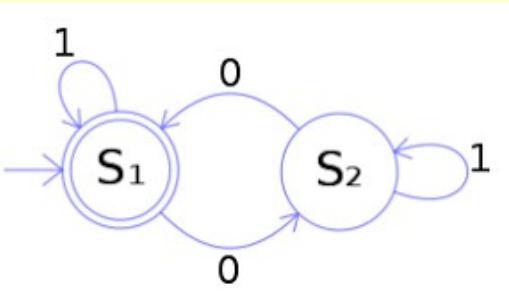
O halts on all inputs iff M halts on w

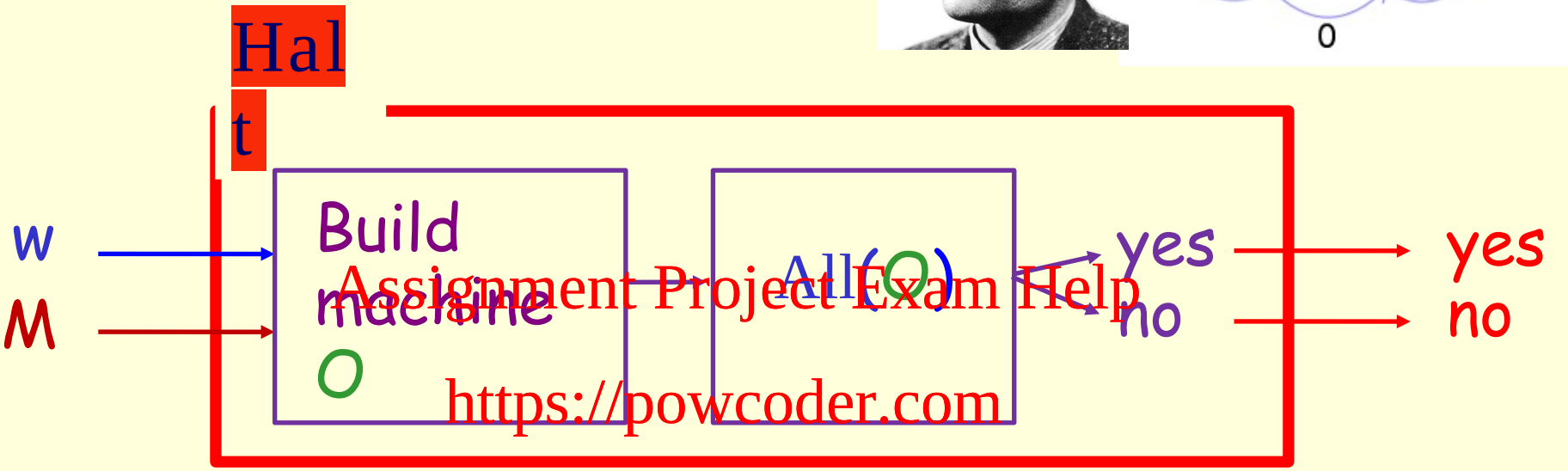
So All on input O will solve the Halting Problem for M on w

So the All Inputs problem is undecidable

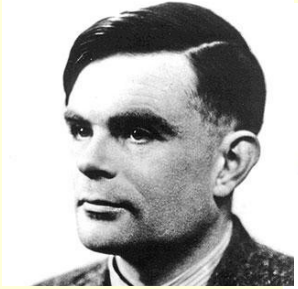
Computability

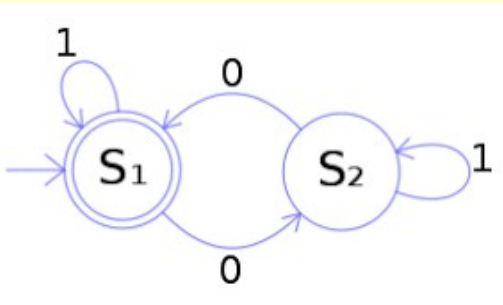






Computability





Undecidable problems include

Halting problem: Give a function f , does it halt on a given input x?
Totality problem: Give a function f , does it halt on every input x?
No input halting problem: Give a function f with no input, does it halt?
Program equivalence: Do two functions f and g always return same value?
Uninitialized variables: Is the variable x initialized before it's used?
Dead code elimination: Does this statement ever get executed?
...
Does $L(G) = \Sigma^*$?
Does $L(G_1) = L(G_2)$?
For PDAs, does $L(M_1) = L(M_2)$?
Does a PDA M have the minimal number of states?

Decidable problems include

Does $L(G) = ?$
Is w $L(G)$?

Questions?

Questions?

Questions?

The Platypus Game

The Platypus Game

Is the halting problem for the Platypus game decidable?

For the Platypus game the halting problem decidable is?

YES!

2^{21} possible tapes
21 x 21 possible head positions
4 x 4 possible states

$2^{21} \times 21 \times 21 \times 4 \times 4 =$ **14,797,504,512**

The Platypus Game

So there are "only" **14,797,504,512** possible configurations of the Platypus game …

P →

Halt-platypus

1. Set counter to 0
2. Run game, incrementing counter after each turn
3. If the game halts, output yes
4. If the counter exceeds 14,797,504,512, output no

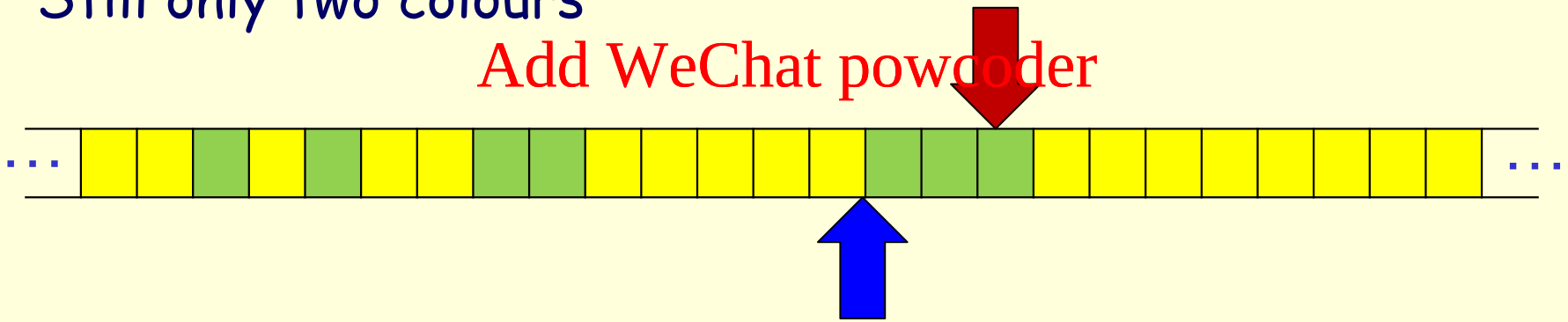
→ yes
→ no

Not very practical, but possible in principle

The Platypus Game

Generalised Platypus game

- Same as Platypus game, but with an infinite tape!
- Ghost gum and Wattle are infinitely far apart
- Arbitrary number of animals
- Still only two colours



Is the Halting problem for the Generalised Platypus game decidable?

The Platypus Game

Survey results (final)

Attempts: 56 out of 70

Which of the following approaches should we use to approximate a tournament of 268,435,456 machines?

"Round robin + knockout" means we divide the machines up into chunks of say 1,000, play a full tournament for each chunk, and then have a knockout round for the 268,000 or so winners.

"Champions League" means we play as many random matches as possible, and use a ranking (ie a ladder in sporting terms) to rank all teams. Once the "season" is over (ie we have no time to play any further matches), the machine that is at the top of the rankings is the winner.

Round-robin + knockout	30 respondents	43 %	✓
Champion's League	25 respondents	36 %	
Something else	1 respondent	1 %	
No Answer	14 respondents	20 %	

The Platypus Game

Survey results (final)

No change at all	20 respondents	39 %
More points for changing green to yellow (or vice-versa)	7 respondents	14 %
Bonus or penalty scores for terminating the game	12 respondents	24 %
Bonus or penalty scores for reaching a tree	12 respondents	24 %
Score multiplied by the number of platypodes in the machine	6 respondents	12 %
Tiebreaker of some sort	18 respondents	35 %
Other		0 %
No Answer	19 respondents	37 %

The Platypus Game

Survey results (final)

Attempts: 47 out of 47

Should there be any rule changes? Please indicate all that you think should apply.

No change at all	33 respondents	**70 %**
Games run on more than one starting set of cells (all green, "chess like", random)	2 respondents	4 %
Shortening or lengthening the maximum number of turns before a game ends (currently 100)	6 respondents	13 %
More players (3 to 8 say)	2 respondents	4 %
Larger or smaller number of cells (eg 11? 31? 101?)	5 respondents	11 %
Different rules for teaching a tree?	1 respondent	2 %
More colours	4 respondents	9 %
More animals	4 respondents	9 %
2-dimensional board	2 respondents	4 %
Machines change somehow during play	5 respondents	11 %
Something else		0 %
No Answer	23 respondents	49 %

The Platypus Game

Survey results (final)

Do you have any other suggestions or comments about tournaments?

Maybe instead of a champion's league with random matchups there are brackets to determine which machine you can fight against (like an elo rating system to determine which players are suitable matches to play with)..

Sounds interesting too!

A rankings system such as used by the Champion's League suggestion is what is used by a majority of online and real world games. Perhaps we can apply what is learned through stimulating a Champion's league to the real world.

Sure! To a real platypus? ☺

The Platypus Game

Survey results (final)

Do you have any other suggestions or comments about tournaments?

Is there any application in creating tournaments that become more ranked then more tournaments that are completed? For example, a single tournament would show how well a set went against each single player, if they were to lose in their round. However, they may have lost to a higher score but a different pairing didn't win to as high a score. That team would progress unfairly.

So completion rate is important?

Champion's league seems like an interesting way to rank the machines! :)

Ok!

The Platypus Game

Survey results (final)

Are there any other suggestions or comments about scoring?

For the tiebreaker, maybe let the match be that reaches the green cell with platypus be the winner.

Interesting --- so we reward the 'bold platypodes'!

I would like a change to the scoring system that allows the player to be rewarded for taking more risk.

Similar, presumably …

Simple is better, and changes to rules would not change the outcome too much.

No change is easy to do!

The Platypus Game

Survey results (final)

Are there any other suggestions or comments about scoring?

If we were to use a ranking system, it makes most sense to include a tiebreaker such that there is no draw.

Sure, although we can rank for draws as well.

If a grid had could turn a token green with a platypus on it, this would be a terminating play. That should garner a penalty.

There may also be a play that has more changes to green than to yellow, which would also be more likely to trigger an end game, and on the flipside the more yellow would mean the player does not terminate at all which is almost cheating, difficult to say without the context of play...

So certain machines should be outlawed?

The Platypus Game

Survey results (final)

Are there any other suggestions or comments about scoring?

Bonus points for reaching the trees.

Ok!

The Platypus Game

Survey results (final)

Are there any other suggestions or comments about rule changes?

If we have a larger number of cells, then there might be some bonus for reaching a tree since it'll be harder to achieve..

Good thought. Presumably the number of cells and tree options should be linked like this.

Shorter games means faster progression, which I'll deal with happily.

A short game is a good game!

Shortening or lengthening the maximum number of turns before a game ends (currently 100),Larger or smaller number of cells (eg 11? 31? 101?)

Sure.

The Platypus Game

Survey results (final)

Are there any other suggestions or comments about rule changes?

Machines changing somehow during play would make it harder to guess the outcome of matches and thus predict what an ideal winner would look like, however would make it much more visually interesting, with emerging strategies we could perhaps study.

Machines change somehow during play

Sure. How would they change?

Shortening or lengthening the maximum number of turns before a game ends (currently 100)

Sure. More responses yet to be analysed!

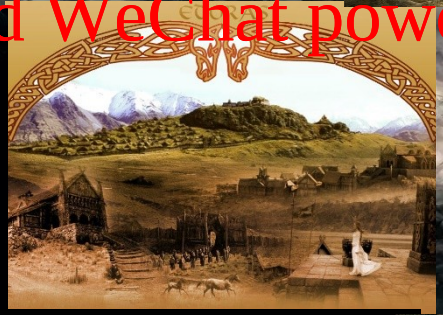
Questions?

Questions?

Questions?


Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Break time! (We resume when all the pictures are gone! This will take 3 minutes!)




Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder
I AM BACK!


Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Alternative Scheme?

Troll	Dreadful	Poor	Acceptable	Exceeds Expectations	Outstanding

Outstanding – CONGRATULATIONS! Your exemplary powers of deduction and a formidable knowledge of the inner workings of the magical world reveal you to be a witch or wizard of genuine skill and learning.

Exceeds Expectations – Well done – a most creditable performance!

Acceptable – demonstrates real magical potential.

Poor – Alas – we regret to inform you that you have narrowly failed. This may have been due to factors outside your control (eg: poltergeist intervention, examination nerves or a malfunctioning quill.) Please do not disconsolate.

Dreadful – We are sorry to inform you that you have failed.

Troll – You would appear either to have abandoned the test due to factors outside your control (eg, earthquake, poltergeist attack), or else you are a troll, in which case you are to be congratulated on being able to use a computer and have achieved the grade of O.F.T. (Outstanding for Trolls).


Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder