cs6501: Great Works in Computer Science **16 January 2013**
David Evans

On Computable Numbers, with an Application to the Entscheidungsproblem

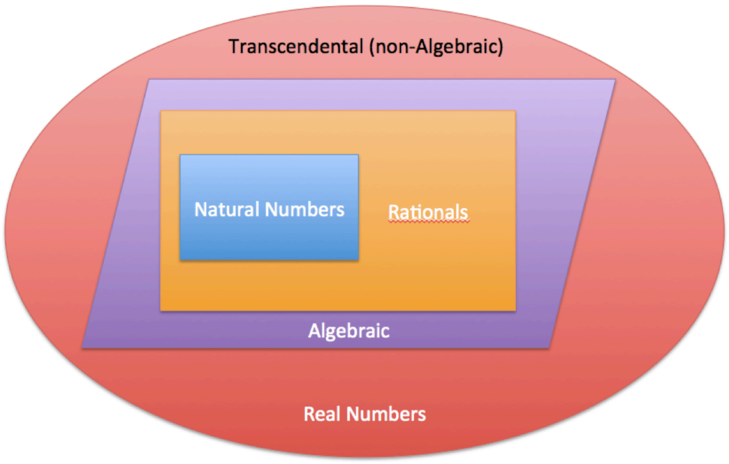
Alan Turing, 1936

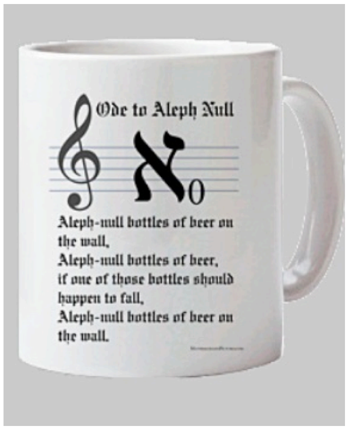
Most alluring would be the attempt at a look into the future and a listing of the problems on which mathematicians should try themselves during the coming century. With such a subject you could have people talking about your lecture decades later.
Hermann Minkowski's advice to **David Hilbert** on his talk to the 1900 Second International Congress of Mathematicians

Alan Turing (1912-1954)

Great ideas introduced in this paper:

1. You can formally model computation.
2. Simple computing model that is still most widely used today.
3. Argument that this model captures everything a computer can do.
4. (Arguably) Subroutines
5. Machine descriptions as first-class objects.
6. Universal machine.
7. Notion of computability
8. "Halting problem"
9. Altenate proof that Entscheidungsproblem is unsolvable: no general decision algorithm
10. Proof of equivalence of TM model and Church's Lambda calculus model.

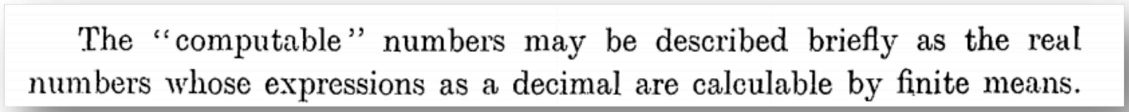



Ode to Aleph Null
$\aleph_0$
Aleph-null bottles of beer on the wall,
Aleph-null bottles of beer,
if one of those bottles should happen to fall,
Aleph-null bottles of beer on the wall.

$\aleph_0$ (aleph null) – cardinality of ennumerable sets

What is the cardinality of the real numbers?

Hilberts #1 Problem: Cantor's *Continuum Hypothesis*: there is no transfinite number between aleph null and $2^{\text{aleph null}}$



The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means.

"I give some arguments with the intention of showing that the computable numbers include all numbers which could naturally be regarded as computable."

a-machine

“*m-configurations*” ~ state of FSM
“tape” divided into “squares”, each can hold one “symbol”
S(r) - symbol on tape square r

Step: erase symbol on tape at tape head position and write a new symbol
change m-configuration
move tape head one square left or right

“It is my contention that these operations includes all those which are used in the computation of a number.”

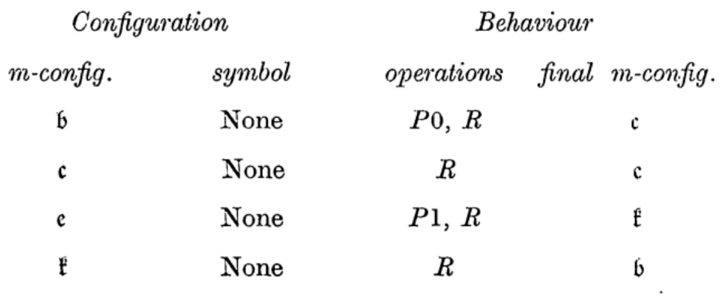
“computing machine” – output figures are sequence of 0 and 1, representing number in binary, number interpreted as binary fraction, $\sum_{n=1}^{\infty} S'(n)2^{-n}$ where $S'(n)$ is the figure on the *n*th figure square.

Termination is Bad!
circular - machine that stops printing output figures: it terminates or keeps running forever without printing any more figures
circle-free – “good machine” - it keeps printing output figures forever

computable number: number whose non-integral part can be produced by a “circle-free” machine

Which of these are computable numbers?
6501
0.25
$^1/_3$
π
...

3. Examples of computing machines.



Configuration		*Behaviour*	
m-config.	*symbol*	*operations*	*final m-config.*
𝔟	None	$P0,\ R$	𝔠
𝔠	None	R	𝔠
𝔢	None	$P1,\ R$	𝔣
𝔣	None	R	𝔟

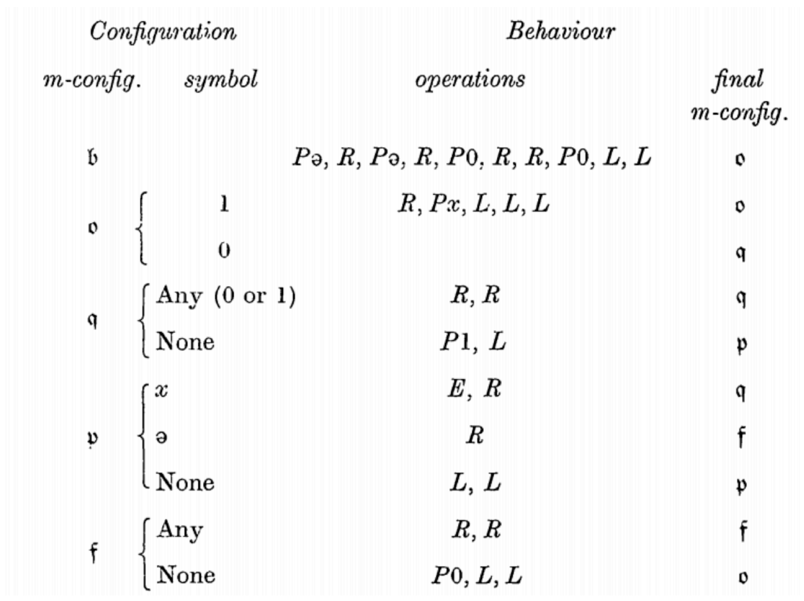
How would you prove this machine is *circle-free*?

Are all rational numbers computable?

Turing's convention: alternate between figure (F-squares) output sequence in binary, write-only) and alternate (E-squares) squares (erasable, any symbols)

Prove this convention doesn't reduce the power of the machine:

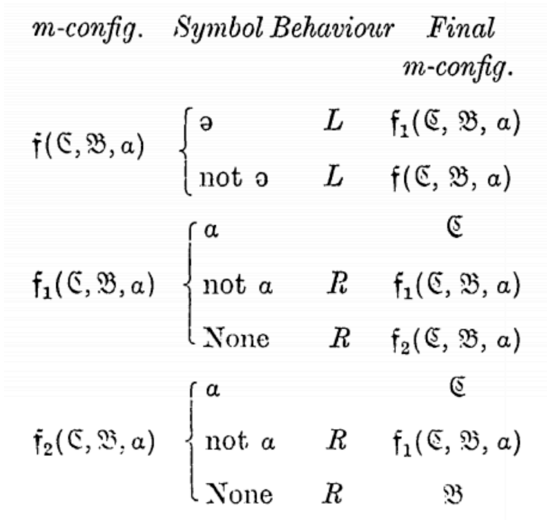
II. "slightly more difficult example" - 001011011101111011111O....



Configuration		*Behaviour*	
m-config.	*symbol*	*operations*	*final m-config.*
𝔟		$P\text{ə}, R, P\text{ə}, R, P0, R, R, P0, L, L$	𝔬
𝔬	1	R, Px, L, L, L	𝔬
	0		𝔮
𝔮	Any (0 or 1)	R, R	𝔮
	None	$P1, L$	𝔭
𝔭	x	E, R	𝔮
	ə	R	𝔣
	None	L, L	𝔭
𝔣	Any	R, R	𝔣
	None	$P0, L, L$	𝔬

4. Abbreviated Tables

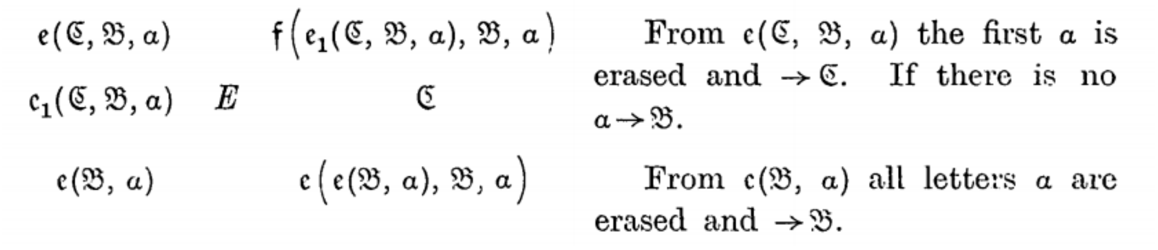
"skeleton tables"



m-config.	*Symbol*	*Behaviour*	*Final m-config.*
$\mathfrak{f}(\mathfrak{C}, \mathfrak{B}, \alpha)$	ə	L	$\mathfrak{f}_1(\mathfrak{C}, \mathfrak{B}, \alpha)$
	not ə	L	$\mathfrak{f}(\mathfrak{C}, \mathfrak{B}, \alpha)$
$\mathfrak{f}_1(\mathfrak{C}, \mathfrak{B}, \alpha)$	α		$\mathfrak{C}$
	not α	R	$\mathfrak{f}_1(\mathfrak{C}, \mathfrak{B}, \alpha)$
	None	R	$\mathfrak{f}_2(\mathfrak{C}, \mathfrak{B}, \alpha)$
$\mathfrak{f}_2(\mathfrak{C}, \mathfrak{B}, \alpha)$	α		$\mathfrak{C}$
	not α	R	$\mathfrak{f}_1(\mathfrak{C}, \mathfrak{B}, \alpha)$
	None	R	$\mathfrak{B}$

Find the bug!

How is this different from a function?

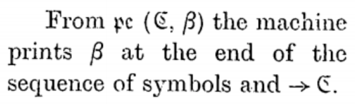


$\mathfrak{e}(\mathfrak{C}, \mathfrak{B}, \alpha)$		$\mathfrak{f}\left(\mathfrak{e}_1(\mathfrak{C}, \mathfrak{B}, \alpha), \mathfrak{B}, \alpha\right)$	From $\mathfrak{e}(\mathfrak{C}, \mathfrak{B}, \alpha)$ the first α is erased and $\rightarrow \mathfrak{C}$. If there is no $\alpha \rightarrow \mathfrak{B}$.
$\mathfrak{e}_1(\mathfrak{C}, \mathfrak{B}, \alpha)$	E	$\mathfrak{C}$	
$\mathfrak{e}(\mathfrak{B}, \alpha)$		$\mathfrak{e}\left(\mathfrak{e}(\mathfrak{B}, \alpha), \mathfrak{B}, \alpha\right)$	From $\mathfrak{e}(\mathfrak{B}, \alpha)$ all letters α are erased and $\rightarrow \mathfrak{B}$.

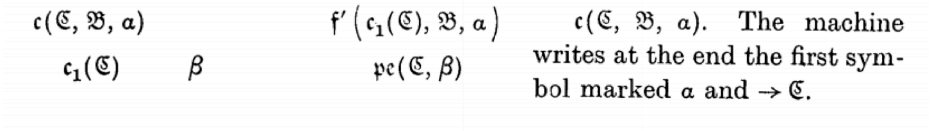
Why isn't this a "recursive" call?

Defining lots of functions useful for building universal machine:

pe(C, b) – print a b at the end of the tape (first blank F-square)

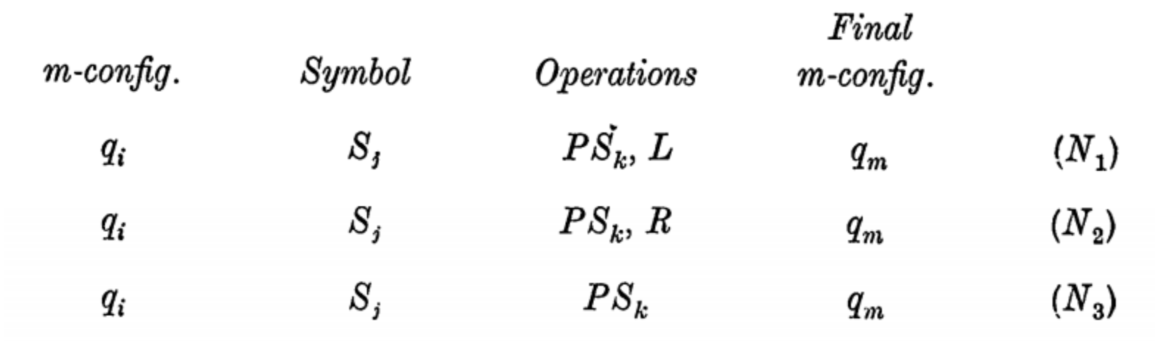

From $\mathfrak{pe}(\mathfrak{C}, \beta)$ the machine prints β at the end of the sequence of symbols and $\rightarrow \mathfrak{C}$.

c(C, B, a) –



$\mathfrak{c}(\mathfrak{C}, \mathfrak{B}, \alpha)$		$\mathfrak{f}'\left(\mathfrak{c}_1(\mathfrak{C}), \mathfrak{B}, \alpha\right)$	$\mathfrak{c}(\mathfrak{C}, \mathfrak{B}, \alpha)$. The machine writes at the end the first symbol marked α and $\rightarrow \mathfrak{C}$.
$\mathfrak{c}_1(\mathfrak{C})$	β	$\mathfrak{pe}(\mathfrak{C}, \beta)$	

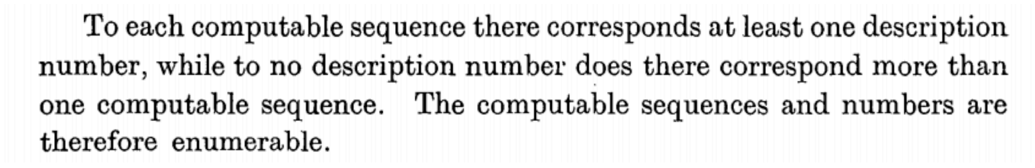
5. Enumeration of computable sequences.



m-config.	*Symbol*	*Operations*	*Final m-config.*	
q_i	S_j	PS_k, L	q_m	(N_1)
q_i	S_j	PS_k, R	q_m	(N_2)
q_i	S_j	PS_k	q_m	(N_3)

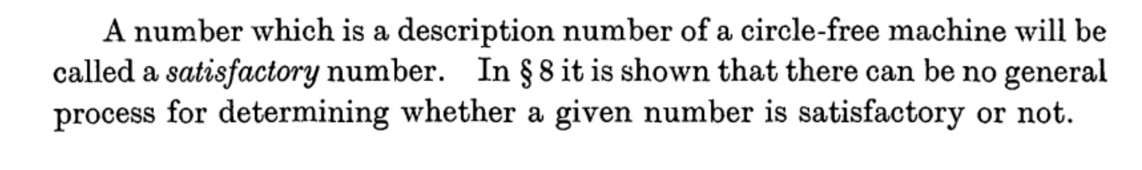
Each table rule: $q_iS_iS_k[L,R,N]q_m$
Convert to string of [ACDLRN]*: $q_k = "DA^k"$ $S_k = "DC^k"$
Convert to number: 123456   7 – separator

$$\mathcal{M}(n) = \text{ the machine described by number } n$$



To each computable sequence there corresponds at least one description number, while to no description number does there correspond more than one computable sequence. The computable sequences and numbers are therefore enumerable.

Obvious/shocking result!



A number which is a description number of a circle-free machine will be called a *satisfactory* number. In § 8 it is shown that there can be no general process for determining whether a given number is satisfactory or not.

6. The universal computing machine.

It is possible to invent a single machine which can be used to compute any computable sequence. If this machine $\mathcal{U}$ is supplied with a tape on the beginning of which is written the S.D of some computing machine $\mathcal{M}$, then $\mathcal{U}$ will compute the same sequence as $\mathcal{M}$.