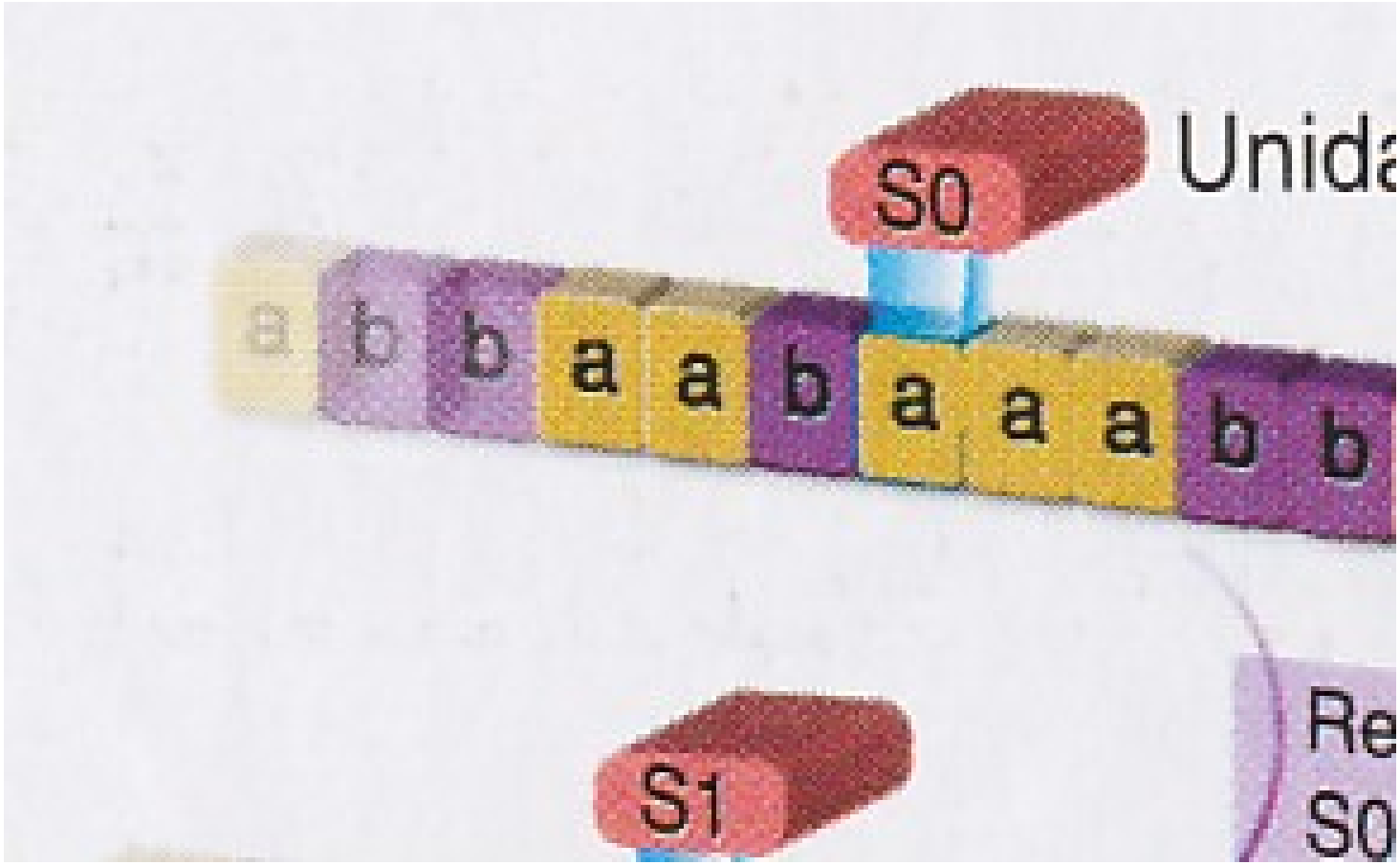
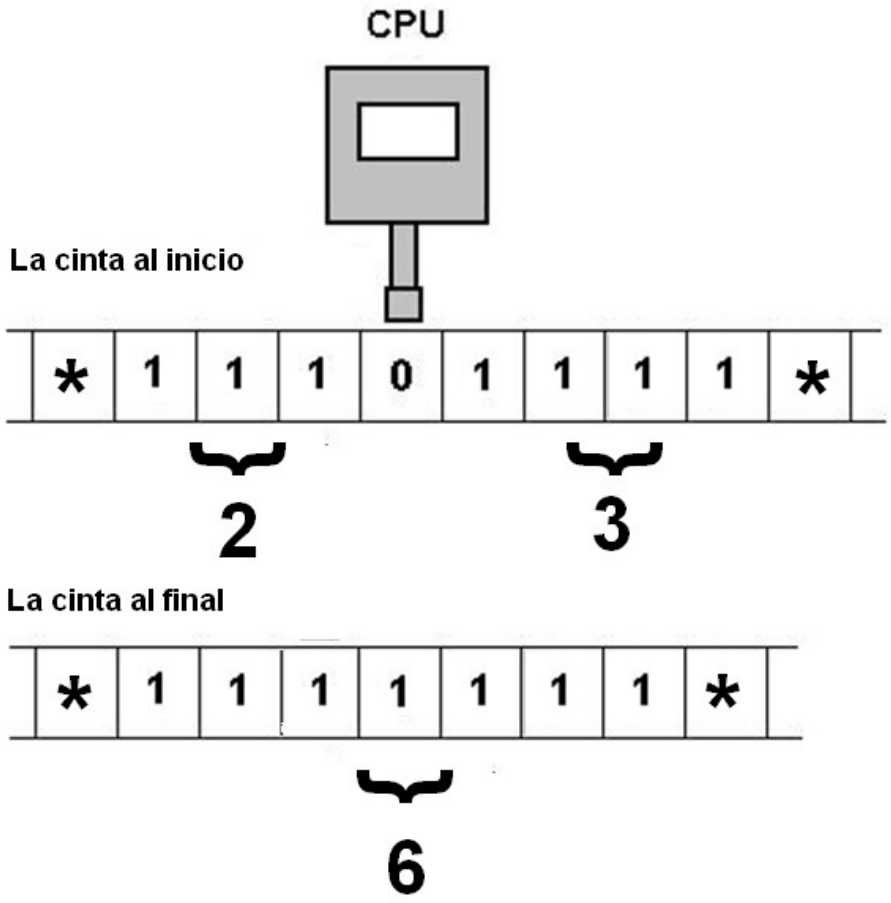
3. Calculando lo incomputable

Pero la contribución científica de Turing consistió en la definición de un "dispositivo" ideal que era capaz de efectuar operaciones muy sencillas pero, por su propia sencillez, podía ser analizado para determinar su capacidad teórica real.

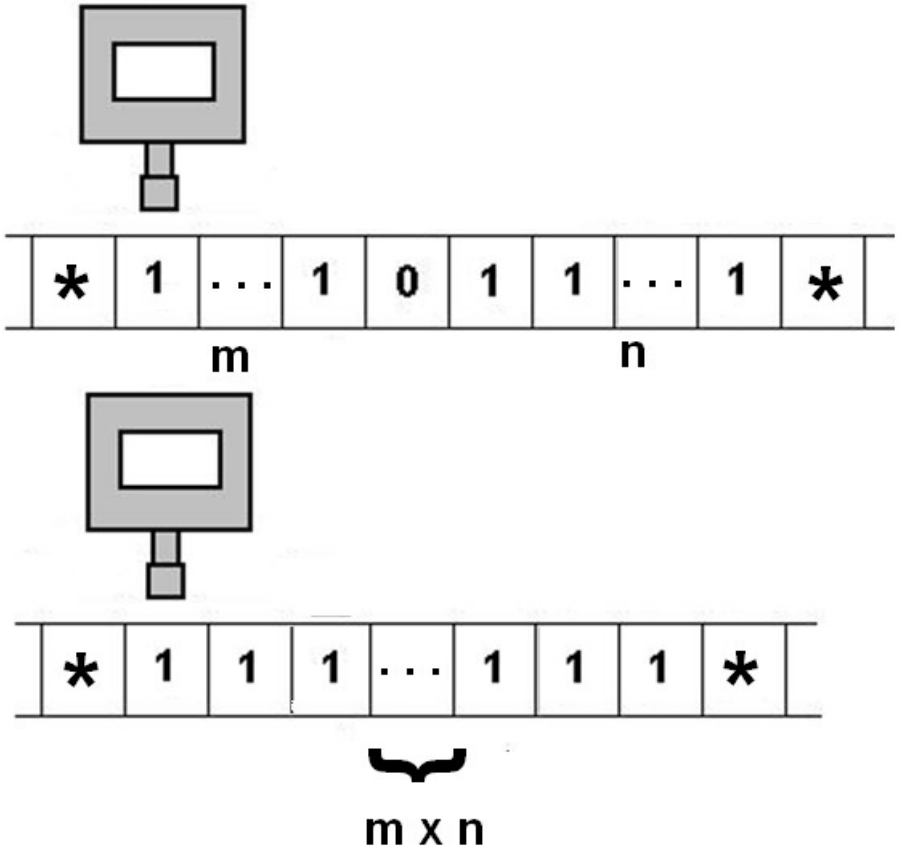
3. Una Máquina de Cómputo



3. Calculando lo Computable



3. Calculando lo Computable



Programa

3. Calculando lo incomputable

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 of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbrous technique. I hope shortly to give an account of the relations of the computable numbers, functions and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.

In §§ 9, 10 I give some arguments with the intention of showing that the computable numbers include all numbers which could naturally be regarded as computable. In particular, I show that certain large classes of numbers are computable. They include, for instance, the real parts of all algebraic numbers, the real parts of the zeros of the Bessel functions, the numbers π, e, etc. The computable numbers do not, however, include all definable numbers, and an example is given of a definable number which is not computable.

Although the class of computable numbers is so great, and in many ways similar to the class of real numbers, it is nevertheless enumerable. In § 8 I examine certain arguments which would seem to prove the contrary. By the correct application of one of these arguments, conclusions are reached which are superficially similar to those of Gödel†. These results

† Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I", *Monatshefte für Math. Phys.*, 38 (1931), 173–198.

3. La Incomputabilidad de **H**

H → El problema del Paro:

"Dada una Máquina de Turing y los contenidos de su cinta ¿Ésta se detendrá?"

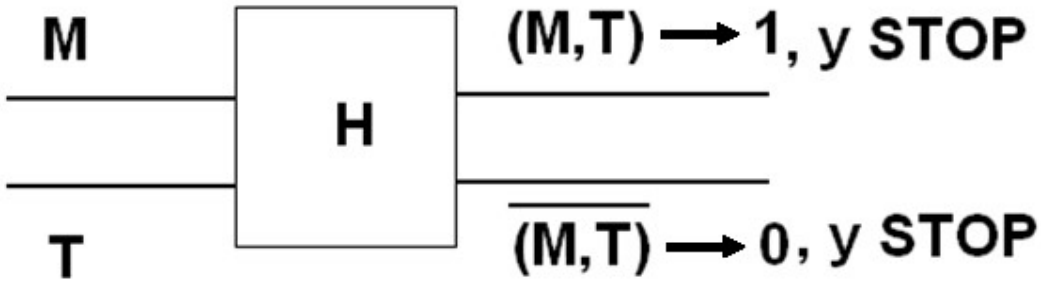
O sea

"Dado un programa y los datos de entrada, el programa terminará en algún momento?"

fermat

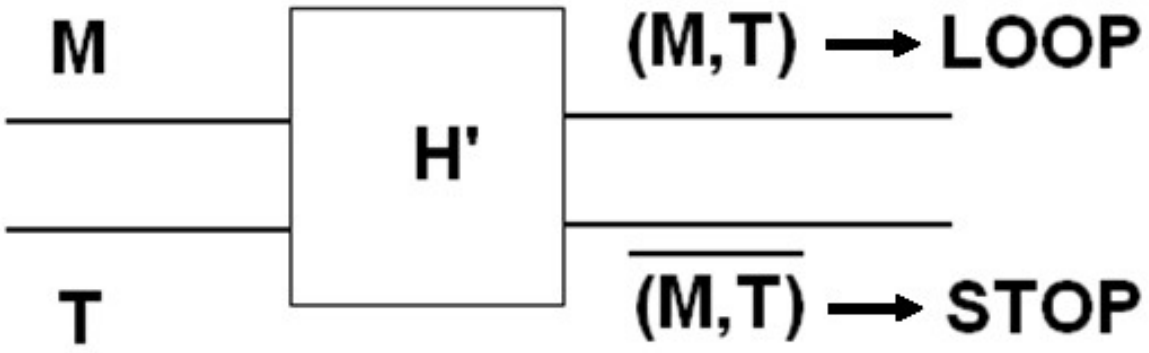
3. La Incomputabilidad de **H**

Supongamos que existe una máquina (llamada “H”) que es capaz de determinar si una computadora (llamada “M”) cuando opera en una cinta (llamada “T”) se detiene, eventualmente.



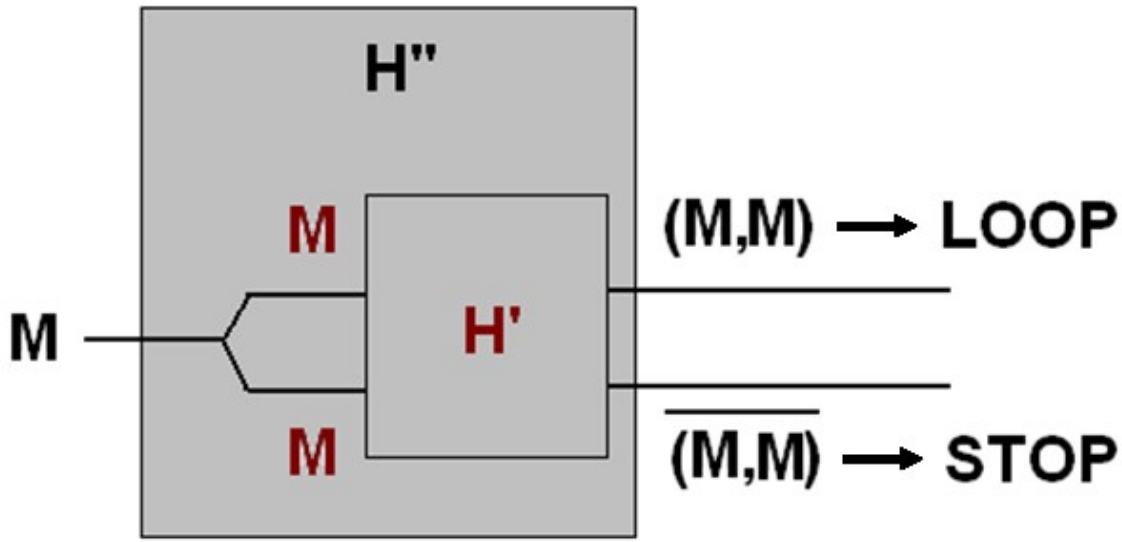
3. La Incomputabilidad de **H**

Es fácil ver que H se puede modificar (y se llama H') para que ahora:

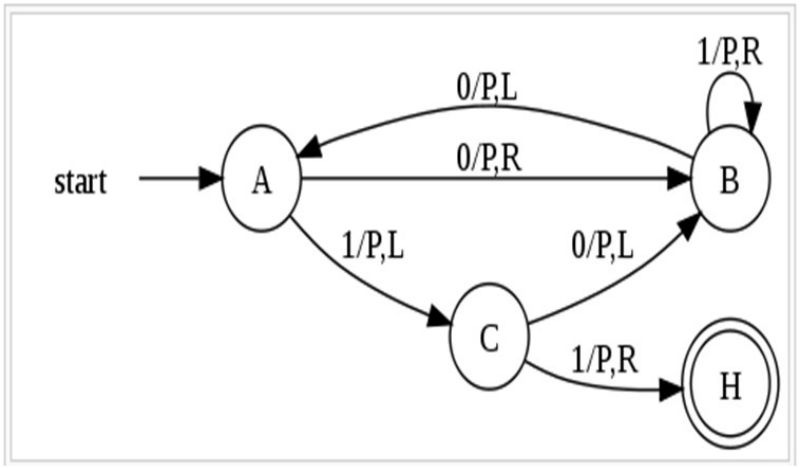


3. La Incomputabilidad de **H**

Ahora modificamos a H' (y la llamamos H'') para que pase lo siguiente:



3a. Apartado: M → M,M



	X=0	X=1
A	1,R,B	1,L,C
B	1,L,A	1,R,B
C	1,L,B	1,R,H

3b. Apartado:

M → M,M

	X=0	X=1
A	1,R,B	1,L,C
B	1,L,A	1,R,B
C	1,L,B	1,R,H

A→ 00
B→ 01
C→ 10
H→ 11

L→ 1
R→0

	X=0	X=1
00	1,0,01	1,1,10
01	1,1,00	1,0,10
10	1,1,01	1,0,11

→ 10011110110010101011011

M

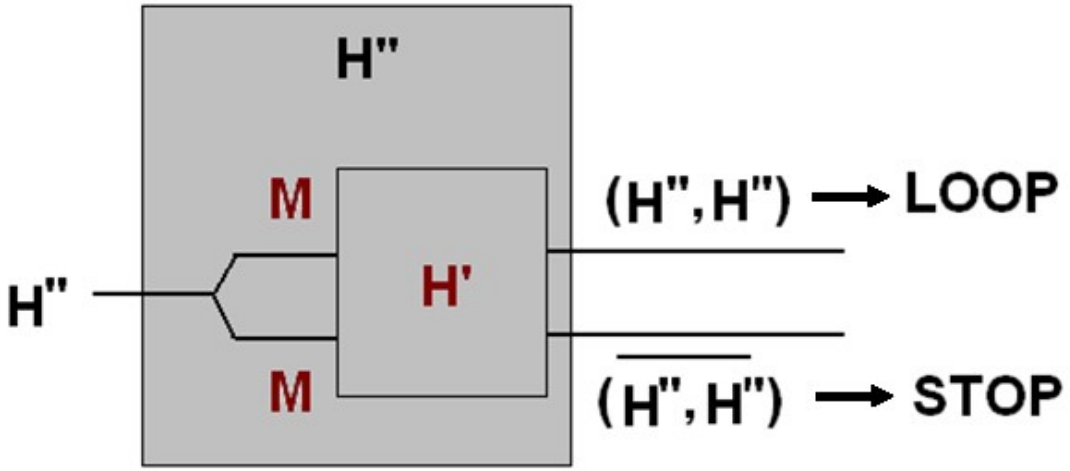
10011110110010101011011 → M 10011110110010101011011

M 10011110110010101011011

3. La Incomputabilidad de **H**

Turing preguntó

¿Qué pasa si a H'' le pasamos, como entrada, los "planos" de ella misma:



3. La Incomputabilidad de **H**

O sea:

$$(H'',H'') \rightarrow \overline{(H'',H'')}$$

$$\overline{(H'',H'')} \rightarrow (H'',H'')$$

Que es equivalente a decir

$$0 = 1$$

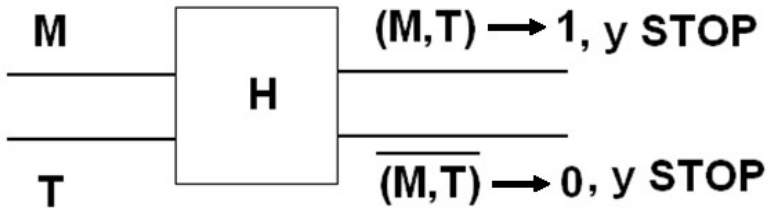
y

$$1 = 0$$

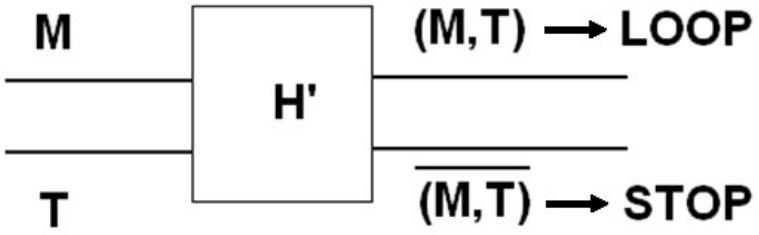
3. La Incomputabilidad de **H**

1

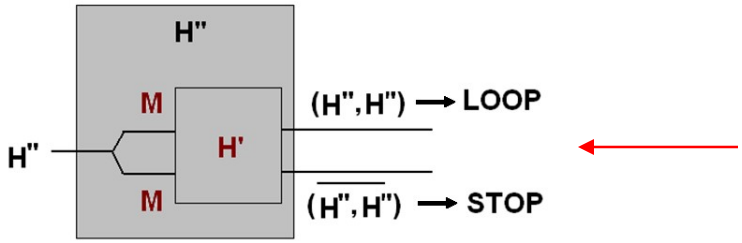
¡Pero cada uno de los pasos anteriores es correcto!



2



3



3. La Incomputabilidad de **H**

La premisa "H EXISTE" es la que induce la contradicción. Es decir

¡H no puede existir!

Lo interesante son las implicaciones.

Funciones Incomputables

Dado que una MT es capaz de hacer todo lo que hace una computadora, Turing nos proveyó de una herramienta que nos permite saber qué podemos computar y que no.

NO ES LO MISMO QUE UN PROBLEMA SEA MUY DIFÍCIL A QUE SEA INCOMPUTABLE.

Problemas MUY Difíciles

Calcula el n-ésimo dígito de Π.

Dime cuál es el 10^{100} –ésimo número primo.

Calcula la posición del sol cuando la edad del universo sea el doble de la actual.

Dado el ADN de un hombre, dime cuánto va a medir cuando cumpla 15 años.

9 Preguntas MUY Difíciles

1. - ¿Cuándo va a parar esta lluvia? (Noé, año 4,314AC)
2. - ¿Cómo se te ocurrió eso? (Su mamá a Pitágoras, 126 AC)
3. - ¿Cómo es que hace este fregado calor? (Juana de Arco, 1,431)
4. - ¿Cuándo vamos a llegar? (Cristóbal Colón, 1,492)
5. - ¿Cómo quieren que pinte este techo? (Miguel Ángel, 1,566)
6. - ¿Pero qué te tomaste Julieta? (Romeo, 1,595)
7. - ¿De dónde salieron todos estos indios? (General Custer, 1,877)
8. - ¿Por dónde entra tanta agua? (Capt. Smith TITANIC, 1,912)
9. - ¿Cómo que no entienden esto? (Profe del ITAM, 2,010)

Funciones Incomputables

1. No es posible determinar si un texto específico será impreso por un programa.
2. No es posible determinar si dos programas son equivalentes.
3. No es posible saber si dos programas computan la misma función.
4. No es posible determinar cuál es el programa más pequeño que calcule una función dada.

Funciones Incomputables

5. No es posible saber cuál es el número más pequeño calculado por una función arbitraria.
6. No es posible saber si se usará una localidad de memoria específica durante una computación.
7. El problema del castor es incomputable.
8. Dado un programa ejecutable, es imposible determinar cuál es su programa fuente.

3. Calculando lo incomputable
El Castor Ocupado

Video MTs

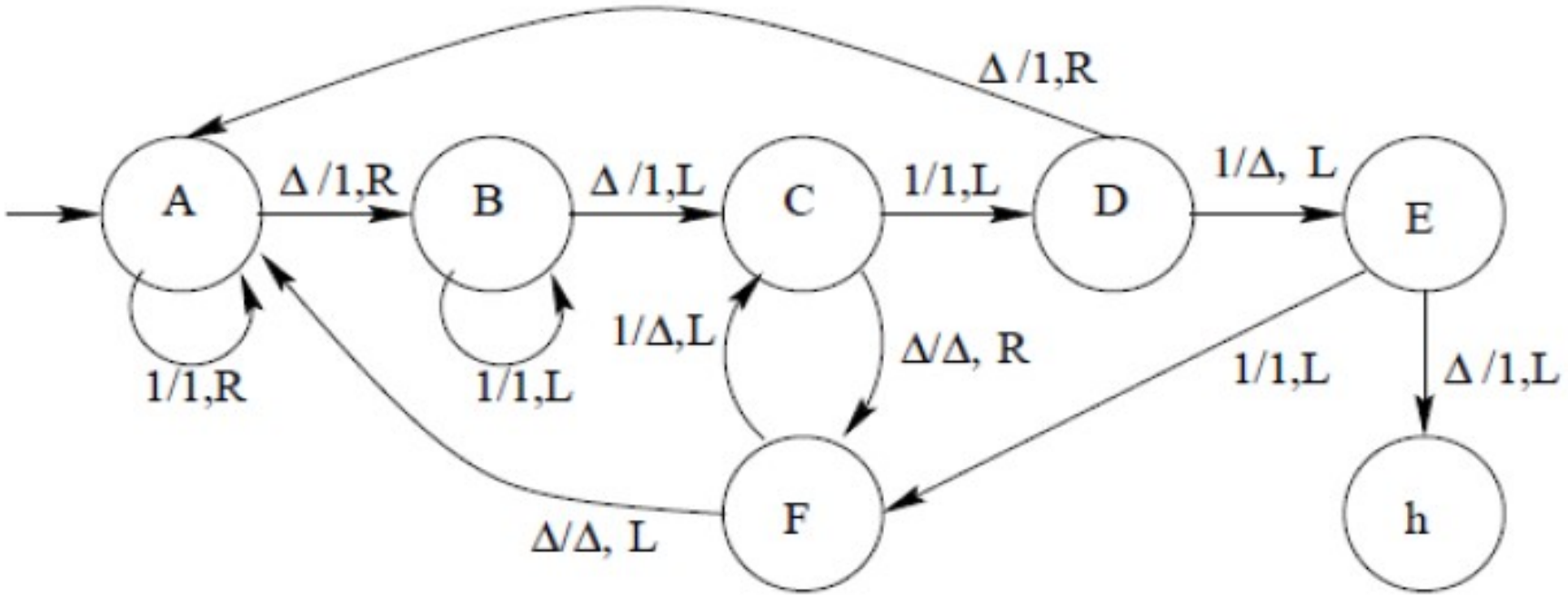
3. Calculando lo incomputable
El Castor Ocupado

La función **Sigma**(n) denota el número máximo de 1s que una Máquina de Turing de dos símbolos y una cinta infinita con n estados puede producir en una cinta inicialmente llena de 0s y detenerse. La función **S**(n) denota el número máximo de pasos que tal MT puede efectuar.

3. Calculando lo incomputable
El Castor Ocupado

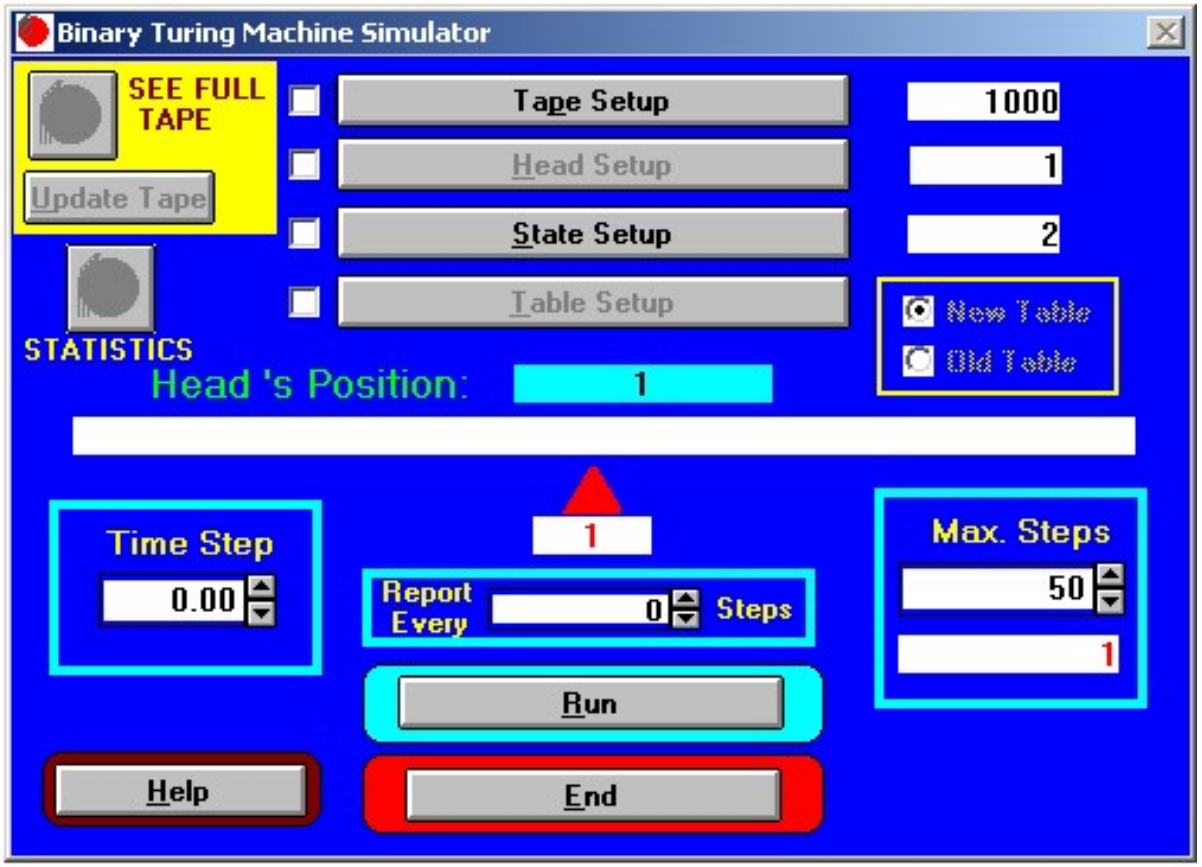
n	$\Sigma(n)$	$S(n)$	Source
1	1	1	Lin and Rado [3]
2	4	6	Lin and Rado [3]
3	6	21	Lin and Rado [3]
4	13	107	Brady [1]
5	≥ 4098	$\geq 47,176,870$	Marxen and Buntrock [4]
6	$\geq 95,524,079$	$\geq 8,690,333,381,690,951$	Marxen [6]

3. Calculando lo incomputable
El Castor Ocupado

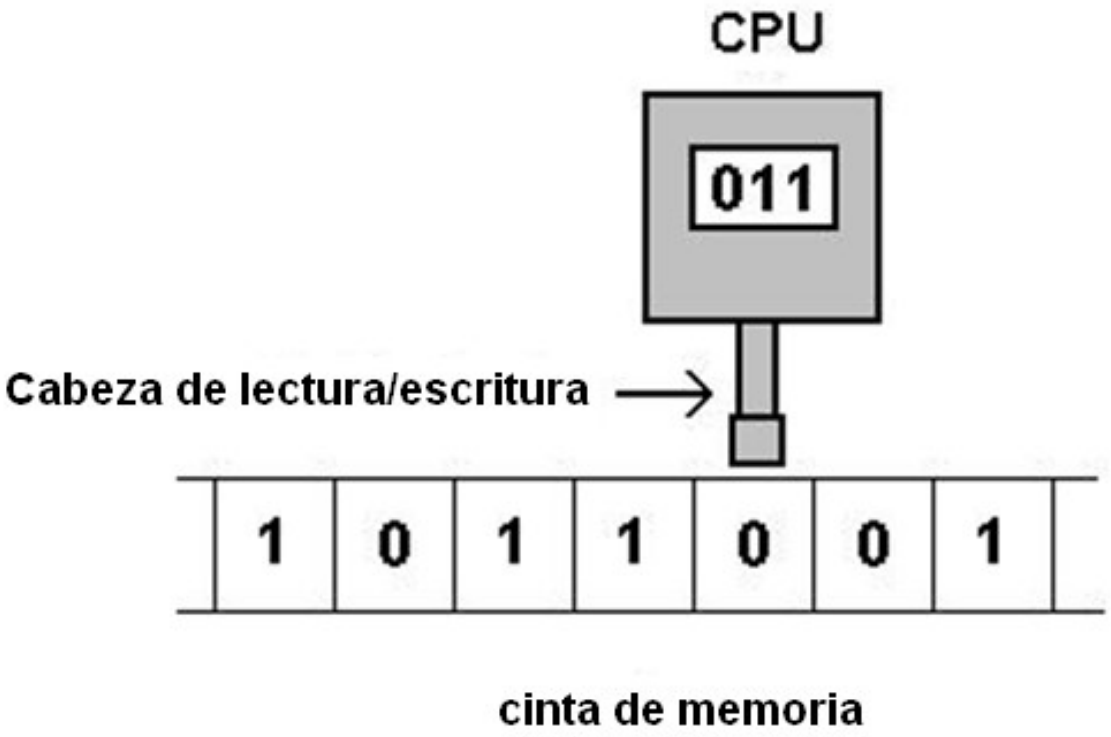


Esta MT escribe 95,524,079 1s y para ello emplea 8,690,333,381,690,951 movimientos.

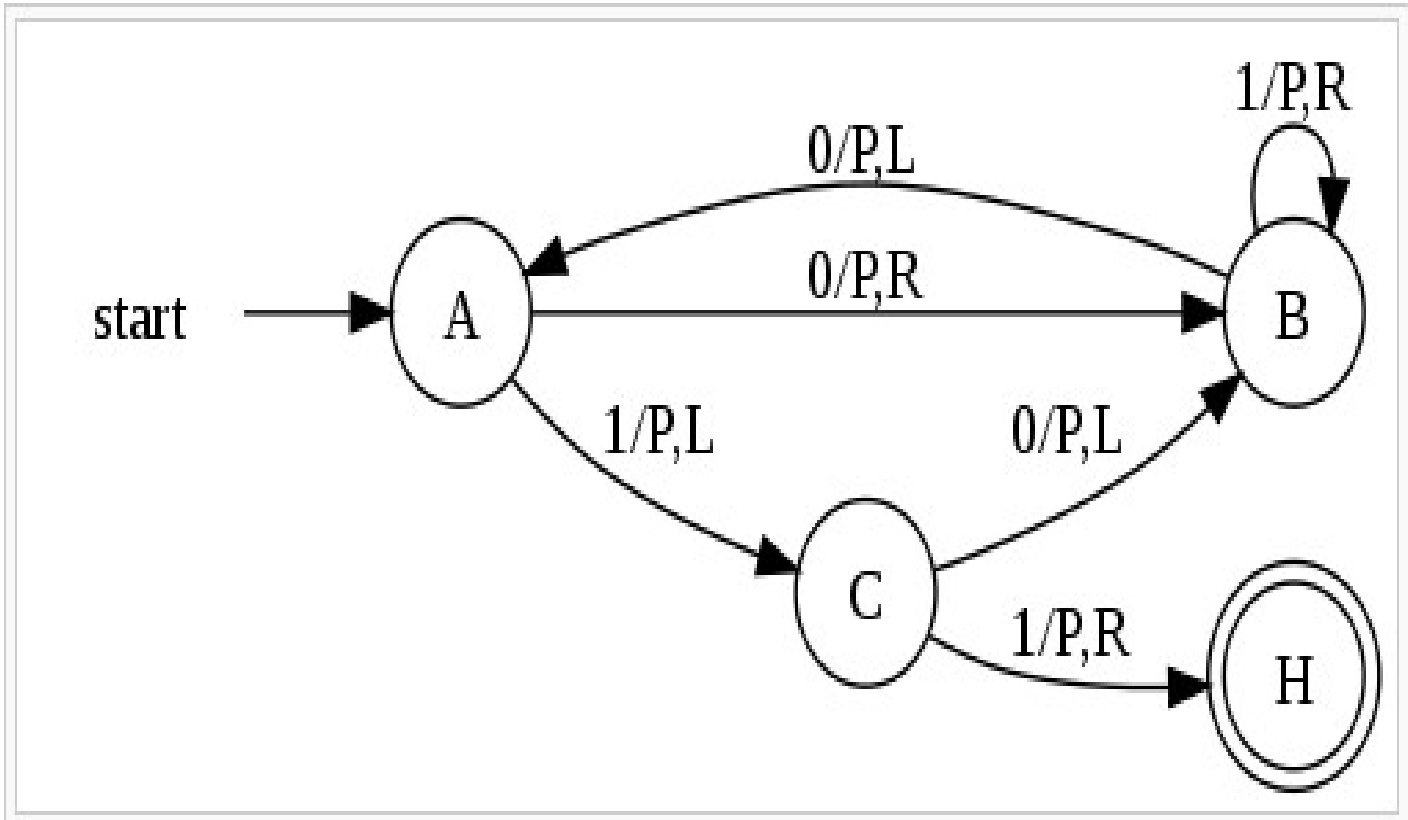
3. Calculando lo incomputable
El Castor de Schult



3. Calculando lo incomputable

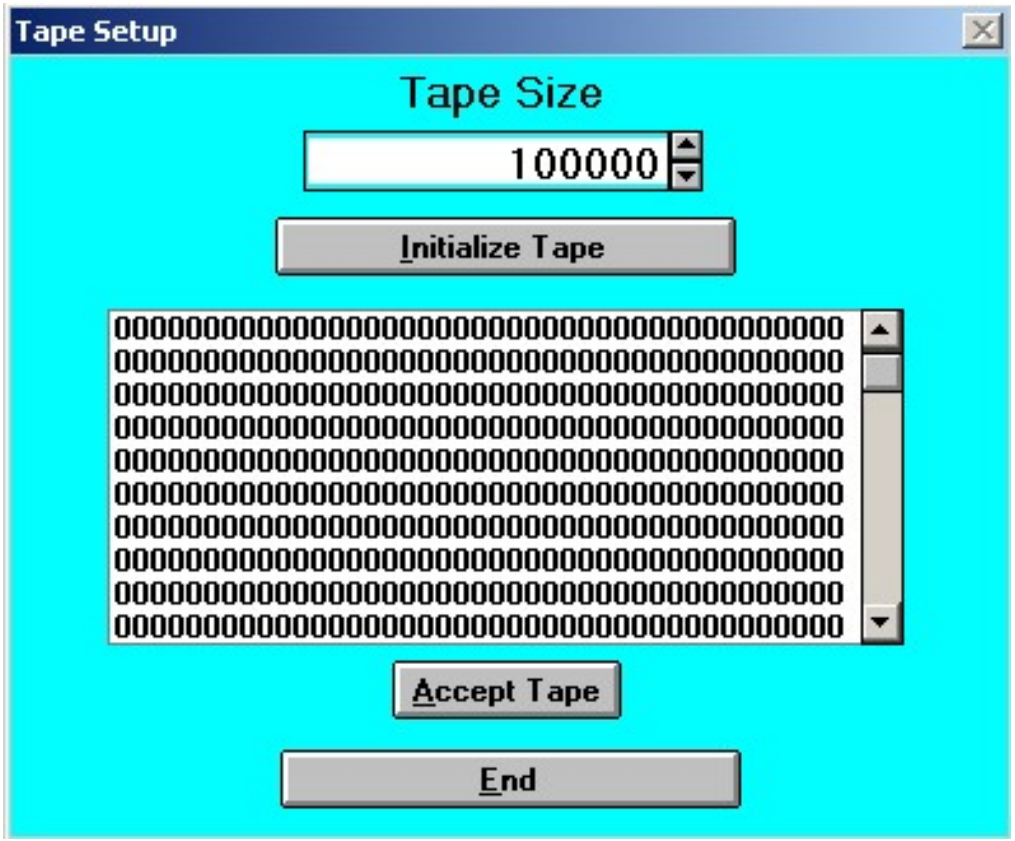


3. Calculando lo incomputable

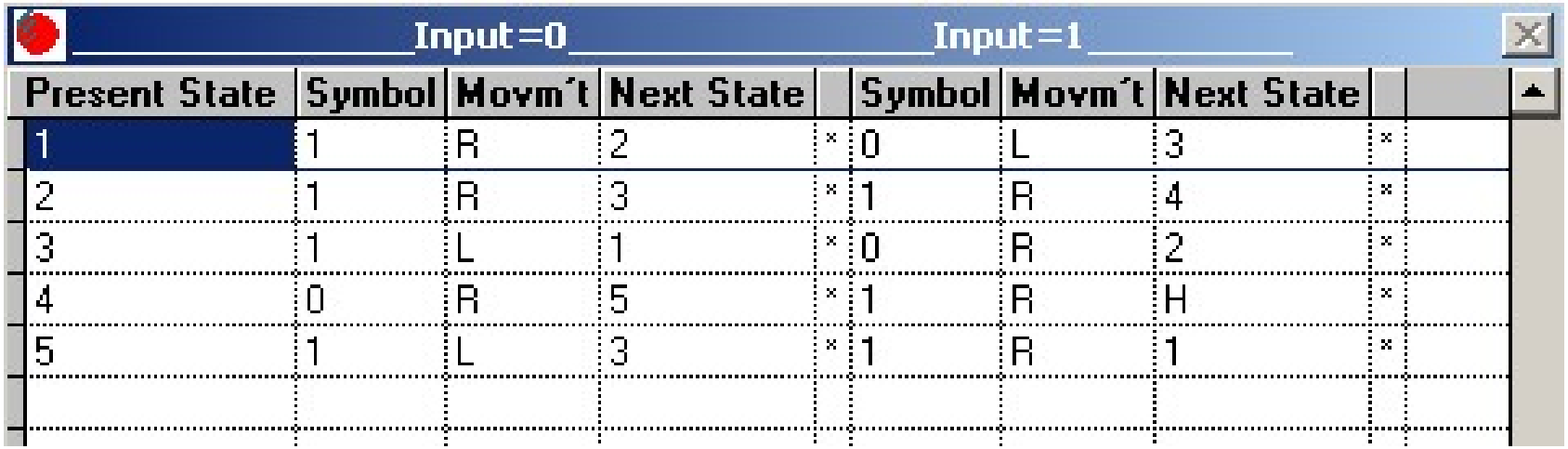


P=1

3. Calculando lo incomputable

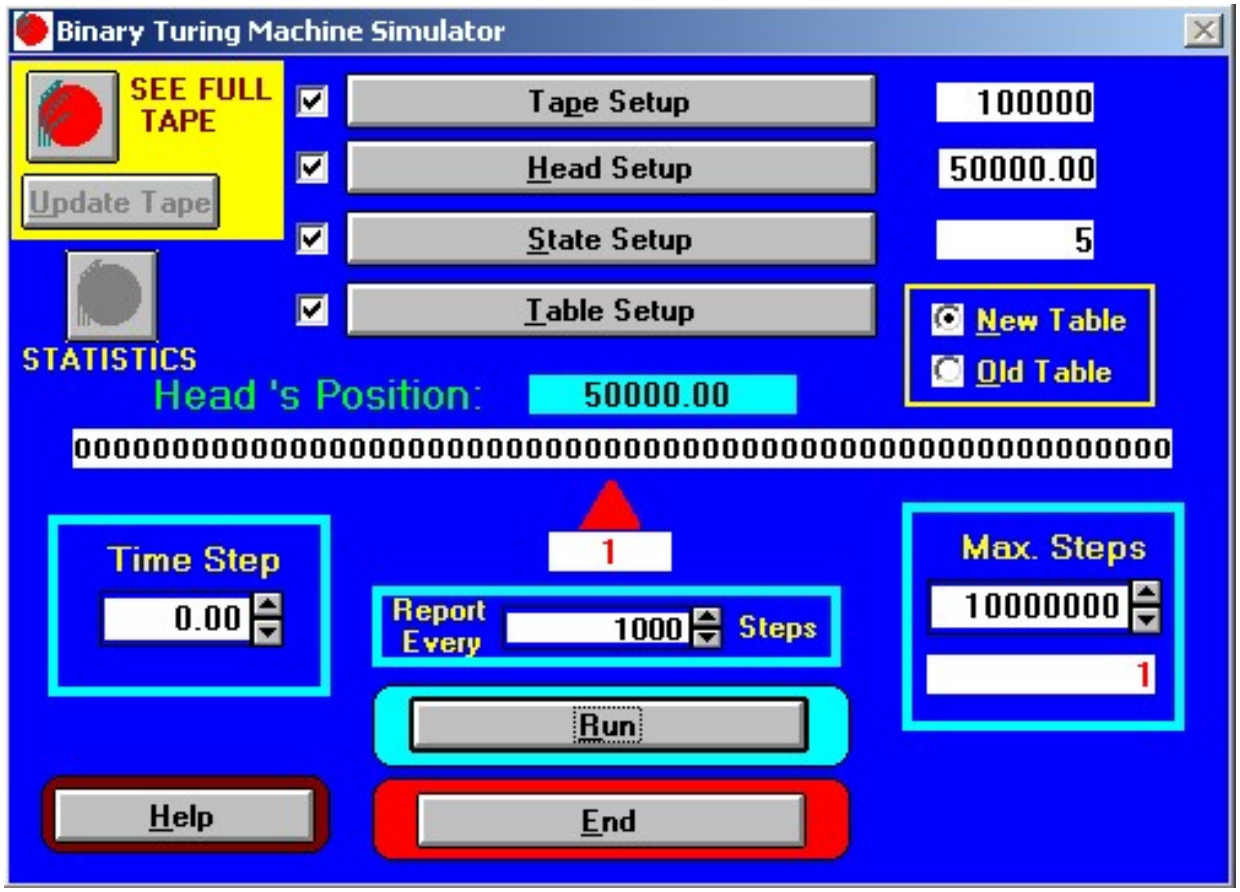


3. Calculando lo incomputable

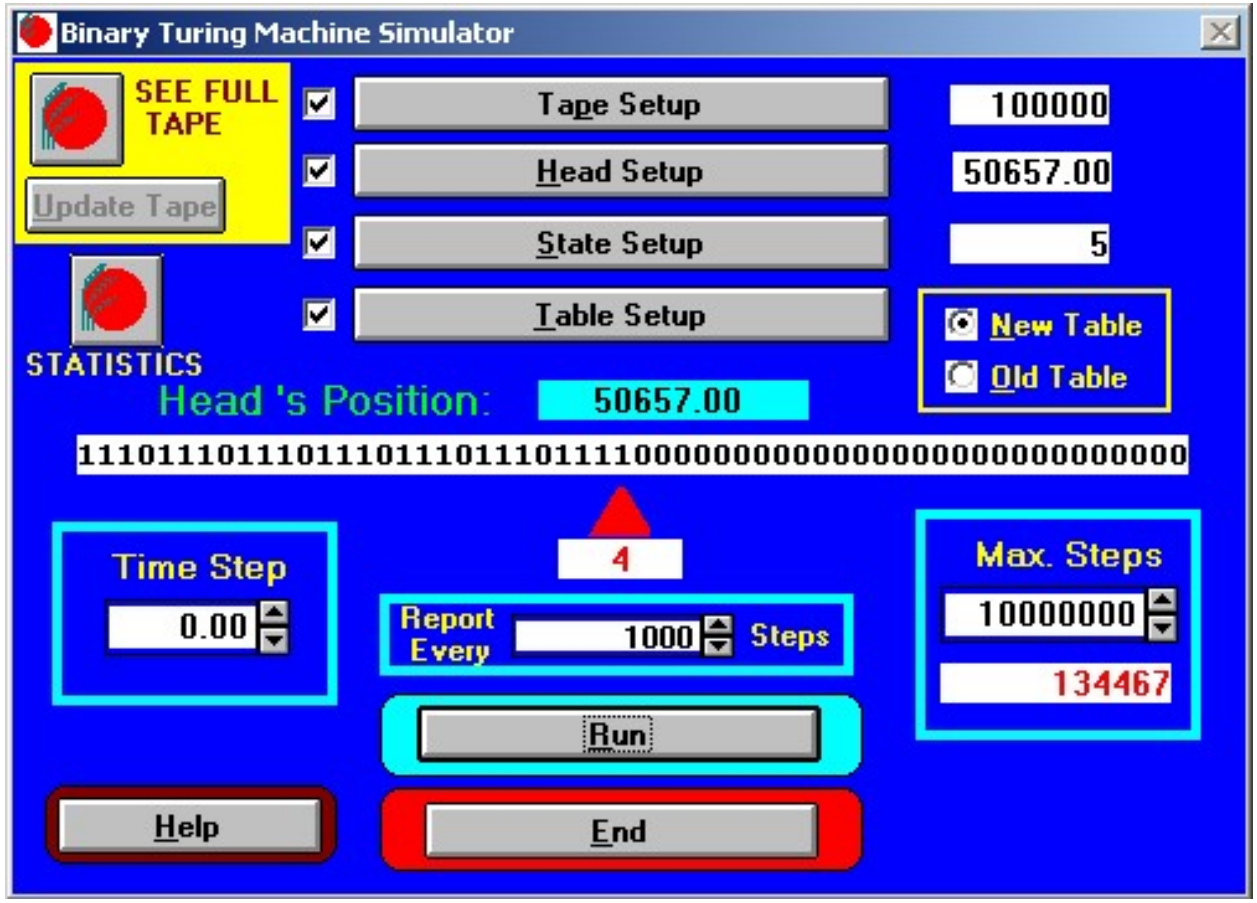


	Input=0				Input=1			
Present State	Symbol	Movm't	Next State		Symbol	Movm't	Next State	
1	1	R	2	*	0	L	3	*
2	1	R	3	*	1	R	4	*
3	1	L	1	*	0	R	2	*
4	0	R	5	*	1	R	H	*
5	1	L	3	*	1	R	1	*

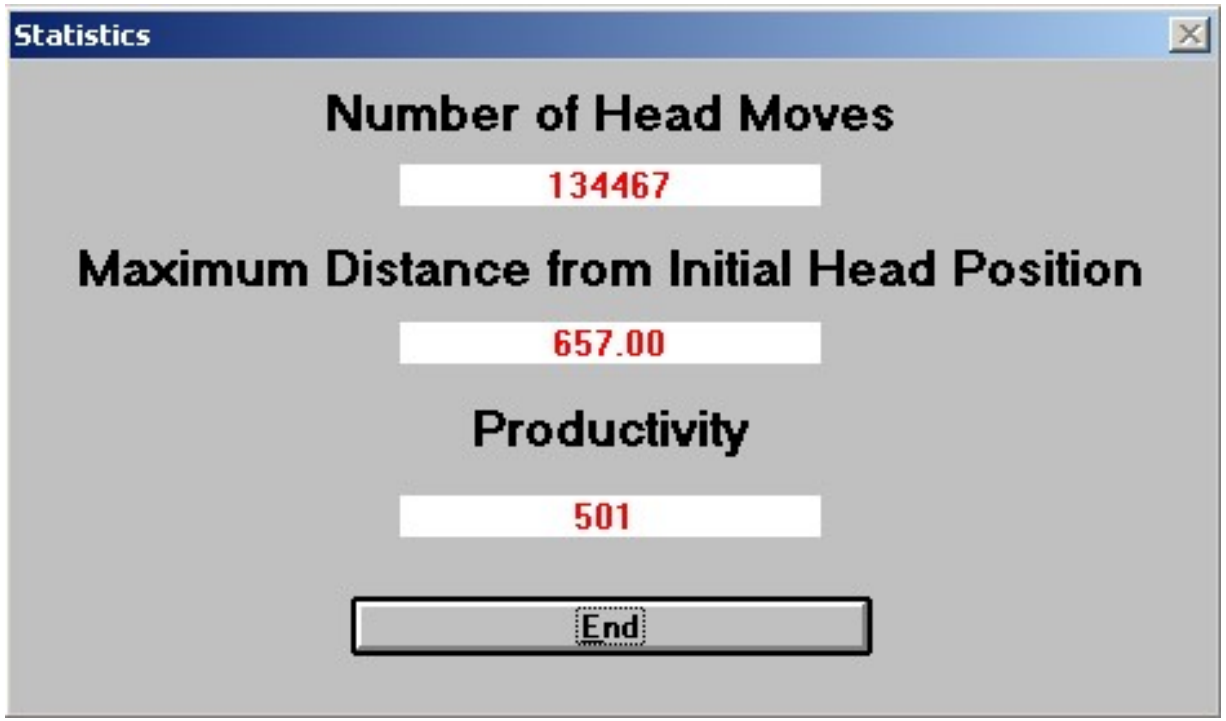
3. Calculando lo incomputable



3. Calculando lo incomputable



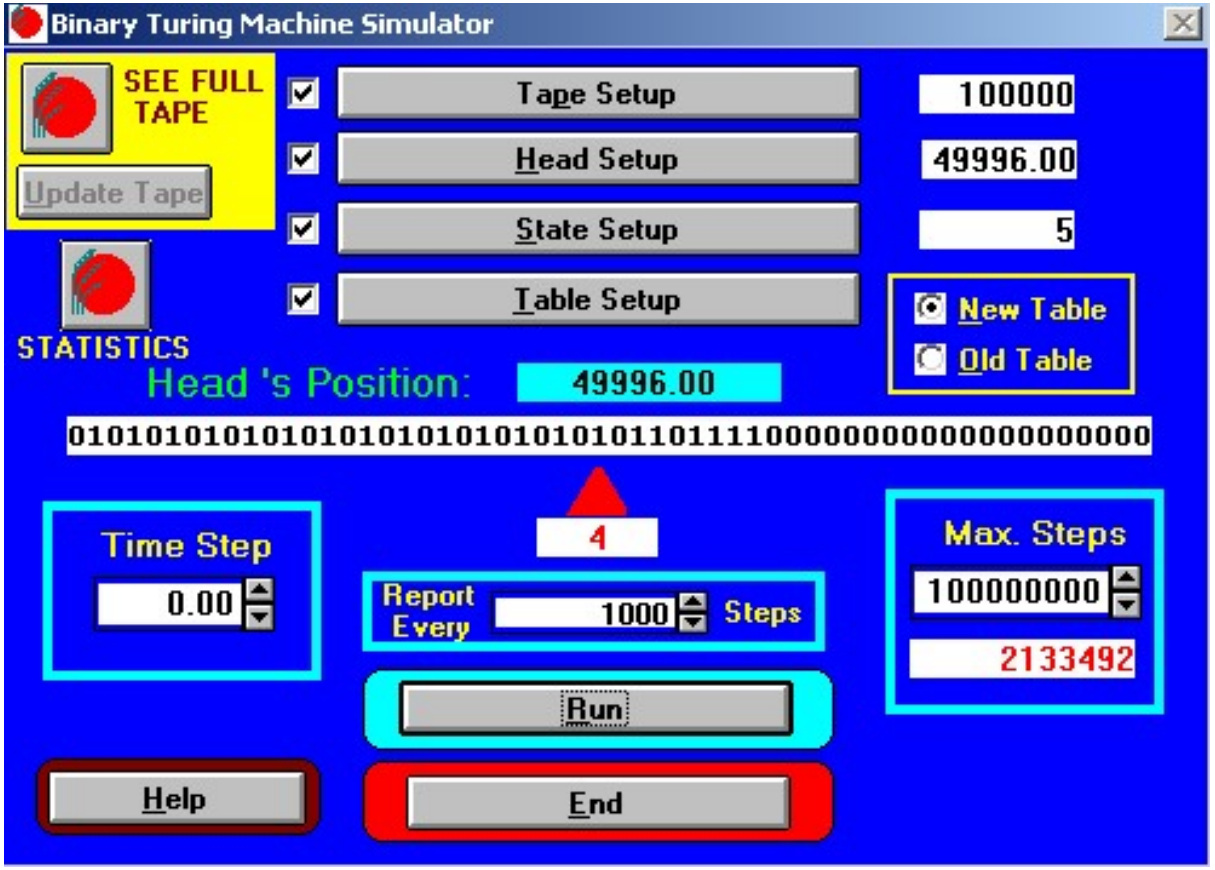
3. Calculando lo incomputable



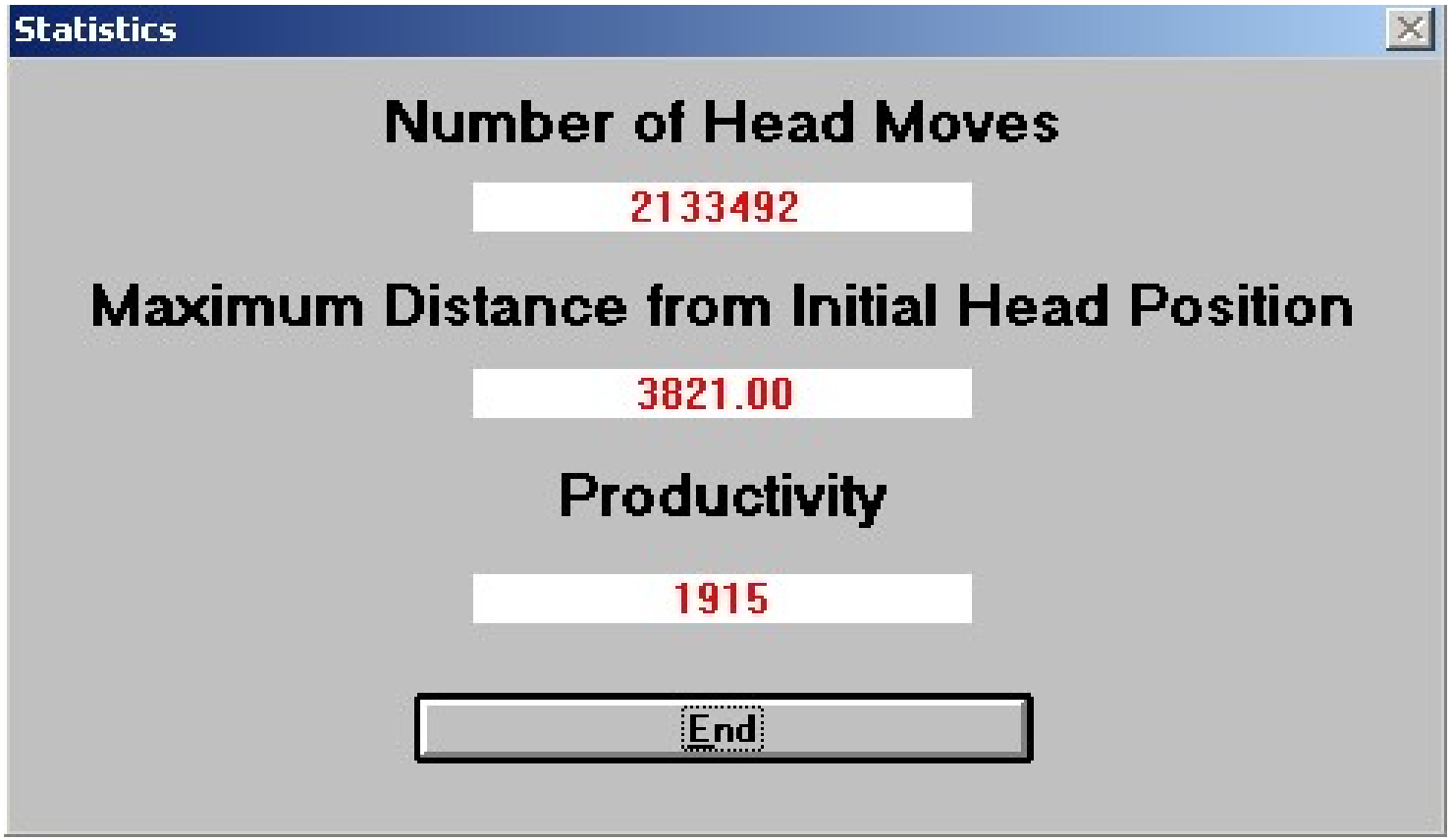
3. Calculando lo incomputable
El Castor de Uhing

	Input=0				Input=1			
Present State	Symbol	Movm't	Next State		Symbol	Movm't	Next State	
1	1	R	2	*	1	L	3	*
2	0	L	1	*	0	L	4	*
3	1	L	1	*	1	L	H	*
4	1	L	2	*	1	R	5	*
5	0	R	4	*	0	R	2	*

3. Calculando lo incomputable



3. Calculando lo incomputable



Video: Castor

Numerando las Funciones Computables

Una vez que se sabe que “Una función computable por una Máquina de Turing” es equivalente a decir “Una función computable”, es posible numerar TODAS las funciones computables.

El número de dichas funciones es, por supuesto, infinito.

Pero ahora podemos asignar a cada una de ellas un “índice”.

Funciones Computables

Las MTs tienen dos tipos de estados:

a) Los estados de paro (H).

b) Los estados activos.

Por ello una MT tiene que tener, al menos, dos estados.

Nos referiremos solamente a las MTs por el número de sus estados “activos”

Funciones Computables
Una MT de un estado

	Símbolo de entrada =0			Símbolo de entrada =1		
Estado Actual	Símbolo de salida	Movimiento de la Cabeza	Siguiente Estado	Símbolo de salida	Movimiento de la Cabeza	Siguiente Estado
A	0	I	A	0	D	H

Digamos que:

D → 0 e I → 1

Los estados se numeran en binario:

0000, 0001, 0010, …, 1110, 1111 y combinaciones.

En hexadecimal:

0, 1, 1, …, E, F

Funciones Computables Numerando las MTs

código 0 0 0 0 0 0

	Símbolo de entrada =0			Símbolo de entrada =1		
Estado Actual	Símbolo de salida	Movimiento de la Cabeza	Siguiente Estado	Símbolo de salida	Movimiento de la Cabeza	Siguiente Estado
A	0	D	A	0	D	A

código 0 0 0 0 0 1

	Símbolo de entrada =0			Símbolo de entrada =1		
Estado Actual	Símbolo de salida	Movimiento de la Cabeza	Siguiente Estado	Símbolo de salida	Movimiento de la Cabeza	Siguiente Estado
A	0	D	A	0	D	H

código 0 0 0 0 1 0

	Símbolo de entrada =0			Símbolo de entrada =1		
Estado Actual	Símbolo de salida	Movimiento de la Cabeza	Siguiente Estado	Símbolo de salida	Movimiento de la Cabeza	Siguiente Estado
A	0	D	A	0	I	A

Funciones Computables
Numerando las MTs

Las MTs anteriores pueden numerarse en hexadecimal: 00,0000b → 00h

00,0001b → 01h

00,0010b → 02h

Así podemos hablar de la MT_0, MT_F, etc.

Funciones Computables
Numerando las MTs

MT_9, por ejemplo, corresponde a

0 0 1 0 0 1

código 0 0 1 0 0 1

	Símbolo de entrada =0			Símbolo de entrada =1		
Estado Actual	Símbolo de salida	Movimiento de la Cabeza	Siguiente Estado	Símbolo de salida	Movimiento de la Cabeza	Siguiente Estado
A	0	D	H	0	D	H

Funciones Computables
Numerando las MTs

Si E=1 $N_1=(2 \times 2 \times 2)^2=64$

Si E=2 $N_2=(2 \times 2 \times 3)^4=20{,}736$

En general

Si E=i $N_i=(2 \times 2 \times (i+1))^{2E}$

Funciones Computables
MTs de 10 estados

$$N_i=(2 \text{ x } 2 \text{ x } (i+1))^{2E}$$

Si E, por ejemplo, es 10, tenemos

$$N_{10} = [(2 \text{ x } 2 \text{ x } (10+1))]^{20}=44^{20}$$

$$\approx 7.3969 \text{ x } 10^{32}$$

$$N_\infty = \sum_{i=1}^{\infty} N_i$$

Funciones Computables (?)
La función *Diagonal*

Definamos ahora la siguiente función (no olvidar que para cada función hay una MT):

$$F_D(X) = F_X(X)+1$$

Por ejemplo,

$$F_D(23) = F_{23}(23)+1$$

$$F_D(1456)=F_{1456}(1456)+1$$

La función *Diagonal*

Dada la lista de MTs (y de funciones) parece fácil implementar $F_D(X)$. Lo único que hay que hacer es ir al renglón "X" de nuestra lista, tomar la MT que está allí, alimentarla con "X", ver qué resultado arroja (si no para suponemos que nos da un "0") y sumarle 1

¡Listo!

La función *Diagonal*

Preguntamos ahora ¿Qué función de nuestra lista infinita corresponde a $F_D(X)$? En esa lista están TODAS las funciones computables.

No puede ser la "1" porque $F_D(1)$ nos da 1+.

No puede ser la "2" porque $F_D(2)$ nos da 1+.

…

No puede ser la "1234567" porque $F_D(1234567)$ nos da 1+.

¡No puede ser **NINGUNA** de nuestra lista!

Funciones Incomputables

$F_D(X)$ no está en nuestra lista y por ello

¡$F_D(X)$ ES INCOMPUTABLE!

Así como construimos $F_D(X)$ podemos construir un número infinito de funciones incomputables haciendo, por ejemplo, $F_{D+2}(X)=F_X(X)+2$ $F_{D+3}(X)=F_X(X)+3$, …

Es decir, existe un infinito de infinitos de funciones no computables.

Funciones Incomputables

Supongamos, ahora, que hacemos una lista infinita de las funciones incomputables.

A las funciones de esa lista las denotamos como F'(X).

Definimos, entonces, la función

$$F'_D(X) = F'_X(X)+1$$

Resulta que $F'_D(X)$ no está en la lista infinita de funciones incomputables.

Funciones Incomputables

Podemos aplicar el mismo procedimiento un número infinito de veces y obtener infinitos de funciones incomputables cada vez más grandes.

Esta noción de "infinitos más grandes" la estableció George Cantor. Al infinito más "chico" le llamó $alef_0$; al siguiente $alef_1$, etc.

Al infinito de funciones computables corresponde $alef_0$. Todos los otros infinitos de funciones nos serán por siempre inalcanzables.

El *alef*

Jorge Luis Borges (no José Luis Borgues) y otros autores estuvieron fascinados por la idea de lo infinito. Por eso (entre otras cosas) escribió “El Alef”, “La Lotería de Babilonia”, “Funes el Memorioso”, etc.

Julio Cortázar, por su parte, escribió “Rayuela”, un libro que no termina nunca.

Y Douglas Hoftstadter escribió “Gödel, Escher and Bach: an Eternal Golden Braid”, un libro que termina antes de terminar.

El *alef*

Pero la noción matemática de lo computable y su enorme jerarquía de infinitos la debemos a la demostración de la equivalencia de “computable” y “calculable por una MT”.