TEMA 8
TEORÍA DE LA COMPUTACIÓN

BIBLIOGRAFÍA
[Dewdney 89]
A. K. Dewdney,
"The New Turing Omnibus: 61 Excursions in Computer Science",
Computer Science Press.
[Eck 95]
David J. Eck,
"The Most Complex Machine",
A. K. Peters.

Introducción

Computabilidad

- En los años 30 se puso en entredicho este concepto.(sgte.)
  - Se planteó la posible existencia de problemas no computables.
  - Se intentó un cambio en la definición de computación.
- Se demostró la equivalencia de [todas] las formulaciones.
- Se demostró la existencia de problemas no computables.

Tesis (conjetura) de Church-Turing:

Función computable
=
máquina de Turing

no hay problema con solución que no sea computable

Alonzo Church

Alan Turing

https://en.wikipedia.org/wiki/David_Hilbert#Hilbert's_program

Hilbert's program [edit]

Main article: Hilbert's program

In 1920 he proposed explicitly a research project (in *metamathematics*, as it was then termed) that became known as Hilbert's program. He wanted mathematics to be formulated on a solid and complete logical foundation. He believed that in principle this could be done, by showing that:

1. all of mathematics follows from a correctly chosen finite system of axioms; and
2. that some such axiom system is provably consistent through some means such as the epsilon calculus.

He seems to have had both technical and philosophical reasons for formulating this proposal. It affirmed his dislike of what had become known as the *ignorabimus*, still an active issue in his time in German thought, and traced back in that formulation to Emil du Bois-Reymond.

This program is still recognizable in the most popular philosophy of mathematics, where it is usually called *formalism*. For example, the Bourbaki group adopted a watered-down and selective version of it as adequate to the requirements of their twin projects of (a) writing encyclopedic foundational works, and (b) supporting the axiomatic method as a research tool. This approach has been successful and influential in relation with Hilbert's work in algebra and functional analysis, but has failed to engage in the same way with his interests in physics and logic.

Hilbert wrote in 1919:

> We are not speaking here of arbitrariness in any sense. Mathematics is not like a game whose tasks are determined by arbitrarily stipulated rules. Rather, it is a conceptual system possessing internal necessity that can only be so and by no means otherwise.[39]

Hilbert published his views on the foundations of mathematics in the 2-volume work Grundlagen der Mathematik.

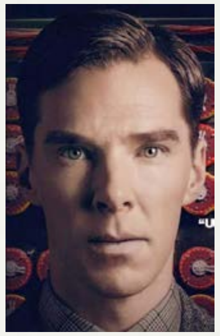
Gödel's work [edit]

Hilbert and the mathematicians who worked with him in his enterprise were committed to the project. His attempt to support axiomatized mathematics with definitive principles, which could banish theoretical uncertainties, ended in failure.

Gödel demonstrated that any non-contradictory formal system, which was comprehensive enough to include at least arithmetic, cannot demonstrate its completeness by way of its own axioms. In 1931 his incompleteness theorem showed that Hilbert's grand plan was impossible as stated. The second point cannot in any reasonable way be combined with the first point, as long as the axiom system is genuinely finitary.

Nevertheless, the subsequent achievements of proof theory at the very least *clarified* consistency as it relates to theories of central concern to mathematicians. Hilbert's work had started logic on this course of clarification; the need to understand Gödel's work then led to the development of recursion theory and then mathematical logic as an autonomous discipline in the 1930s. The basis for later theoretical computer science, in the work of Alonzo Church and Alan Turing, also grew directly out of this 'debate'.

Una “curiosidad” sobre Turing, después de que éste nos contara casi todo.
(de “aquella manera”)



ATHLETICS

MARATHON AND DECATHLON CHAMPIONSHIPS

The Amateur Athletic Association championships for this year were concluded at Loughborough College Stadium, Leicestershire, on Saturday, with the second, and last, day of the Decathlon and the decision of the Marathon championship.

MARATHON CHAMPIONSHIP (26 miles 385 yds.) (record: 2hrs. 30min. 57 6sec., by H. W. Payne, Windsor to Stamford Bridge, on July 5, 1929; standard time: 3hrs. 5min.).—J. T. Holden (Tipton Harriers), 2hrs. 33min. 20 1-5sec., 1; T. Richards (South London Harriers), 2hrs. 36min. 7sec., 2; D. McNab Robertson (Maryhill Harriers, Glasgow), 2hrs. 37min. 54 3-5sec., 3; J. E. Farrell (Maryhill Harriers), 2hrs. 39min. 46 2-5sec., 4; Dr. A. M. Turing (Walton A.C.), 2hrs. 46min. 3sec., 5; L. H. Griffiths (Reading A.C.), 2hrs. 47min. 50 2-5sec., 6.

DECATHLON CHAMPIONSHIP.—H. J. Moesgaard-Kjeldsen (Polytechnic Harriers, London), 5,965 points, 1; Captain H. Whittle (Army and Reading A.C.), 5,650, 2;

La máquina de Turing

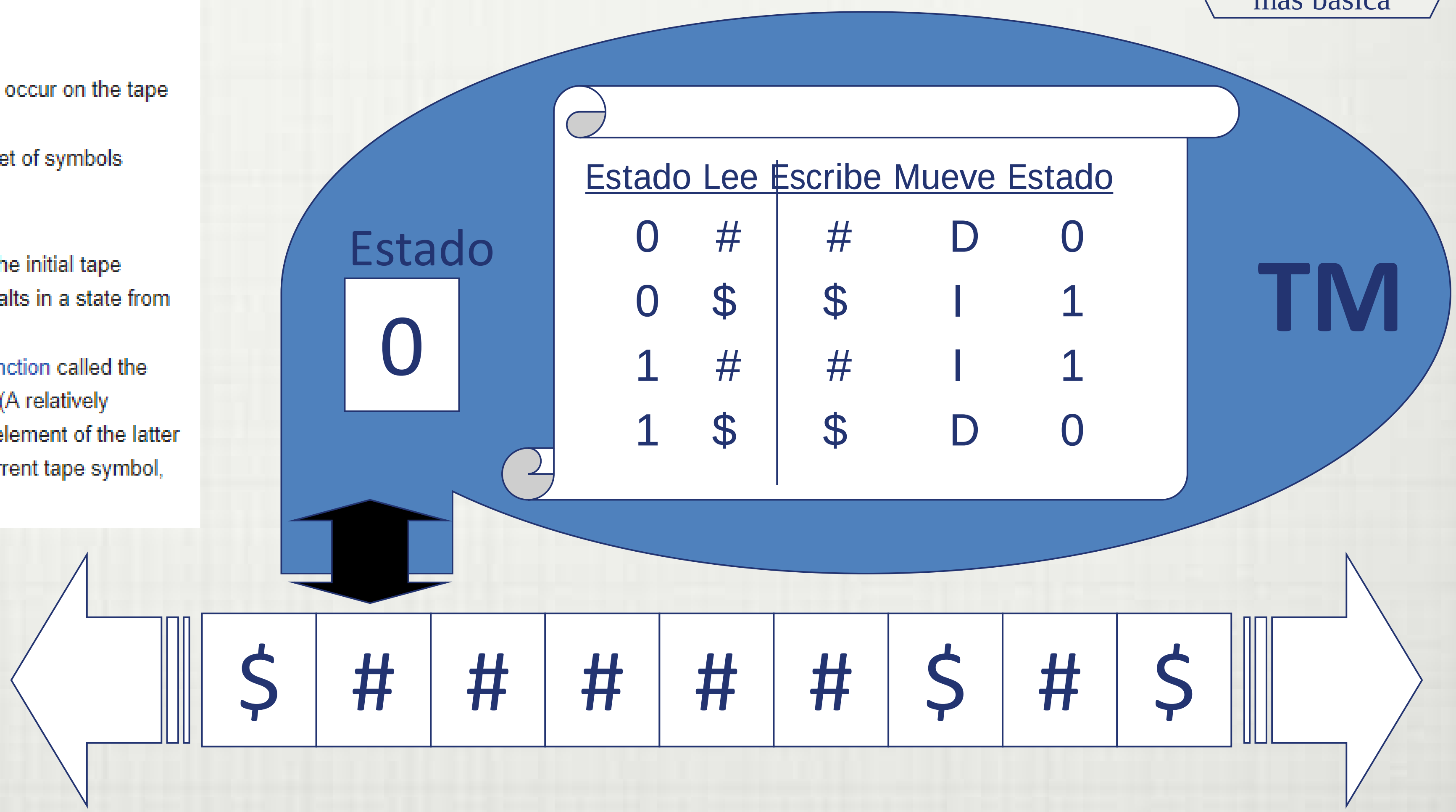
Es la formulación más simple y gráfica del concepto computabilidad.

No es difícil ver una máquina de Turing en un computador cuando lo vemos desde su arquitectura más básica

Formal definition [edit]

Following Hopcroft and Ullman (1979, p. 148)[citation needed], a (one-tape) Turing machine can be formally defined as a 7-tuple $M = \langle Q, \Gamma, b, \Sigma, \delta, q_0, F \rangle$ where

- Q is a finite, non-empty set of *states*;
- Γ is a finite, non-empty set of *tape alphabet symbols*;
- $b \in \Gamma$ is the *blank symbol* (the only symbol allowed to occur on the tape infinitely often at any step during the computation);
- $\Sigma \subseteq \Gamma \setminus \{b\}$ is the set of *input symbols*, that is, the set of symbols allowed to appear in the initial tape contents;
- $q_0 \in Q$ is the *initial state*;
- $F \subseteq Q$ is the set of *final states* or *accepting states*. The initial tape contents is said to be *accepted* by M if it eventually halts in a state from F.
- $\delta : (Q \setminus F) \times \Gamma \nrightarrow Q \times \Gamma \times \{L, R\}$ is a partial function called the *transition function*, where L is left shift, R is right shift. (A relatively uncommon variant allows "no shift", say N, as a third element of the latter set.) If δ is not defined on the current state and the current tape symbol, then the machine halts;[21]



La máquina de la figura no resuelve ningún problema. Ejercicio: diseñar una maquina de Turing que resuelva el problema de fijar paridad par a una secuencia de ceros y unos en la cinta, añadiendo un cero o uno a la izquierda.

La UTM (Universal Turing Machine)

La Máquina Universal de Turing es una máquina de Turing cuya capacidad consiste en simular cualquier máquina de Turing.

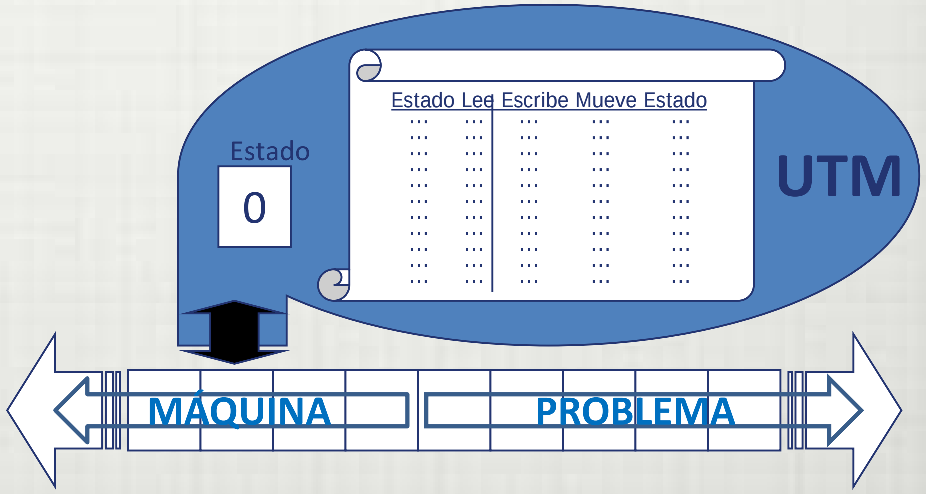
El concepto de completitud de Turing

Trabaja con la cinta dividida en dos secciones que contienen:

- La descripción de la máquina a simular
- La entrada “problema” que debe resolver la máquina simulada.

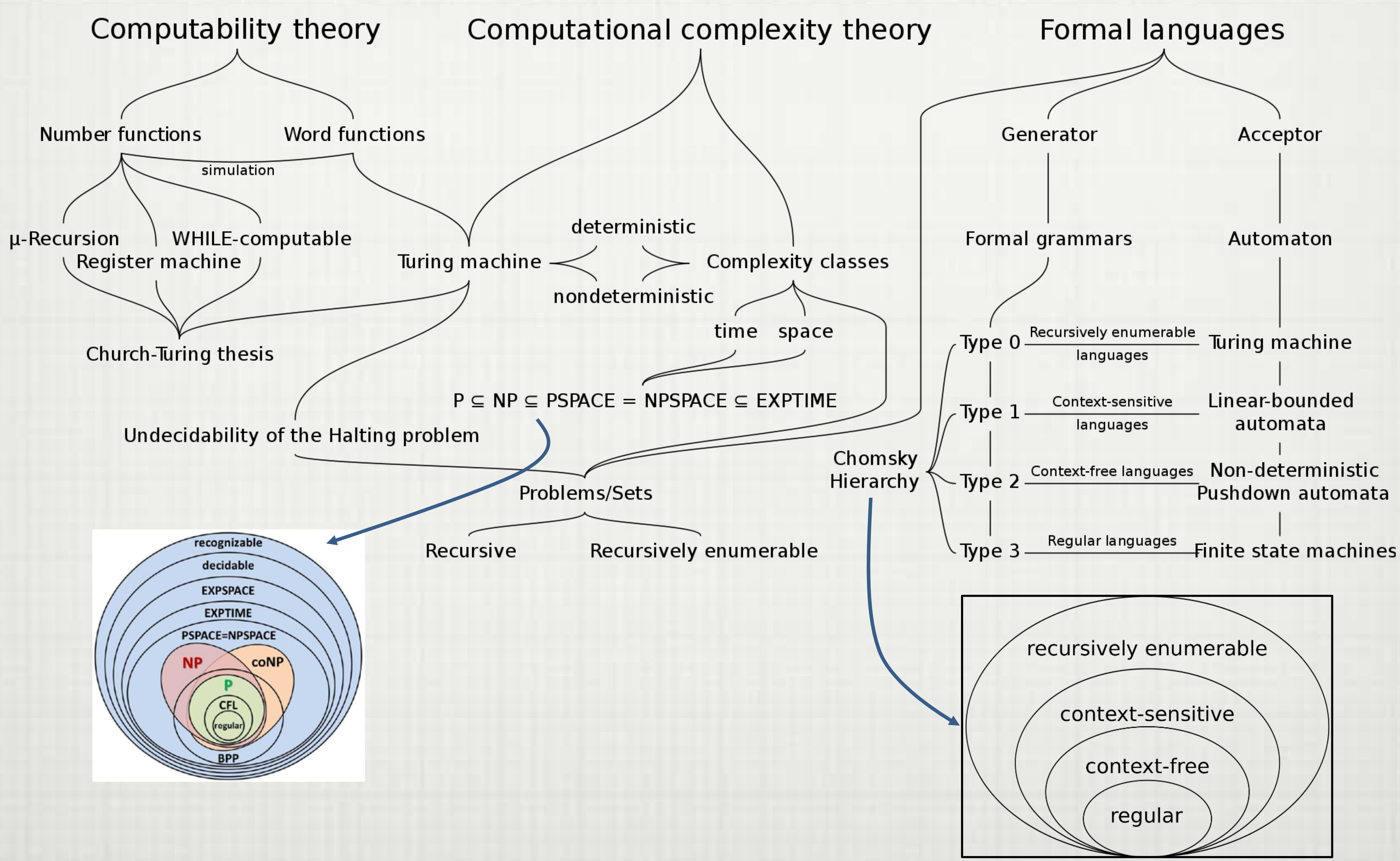
Su tabla expresa dos fases de funcionamiento

- Dado el estado y símbolo de entrada, localizar la **quíntupla** correspondiente en la descripción de la máquina.
- Ejecutar en función de la descripción, sobre la cinta problema.



Complejidad computacional

http://gpd.sip.ucm.es/rafa/docencia/mtp/complejidad/complejidad.html



https://en.wikipedia.org/wiki/Computational_complexity_theory

Problemas no computables

¿Podemos determinar a priori si una maquina <u>cualquiera</u> de Turing terminará por resolver un problema cualquiera que le presentemos? **(Problema de la parada-Halting Problem)**

O, de modo equivalente,…

- ¿parará al ponerla en marcha frente a la cinta vacia?
- ¿resolverá al menos un problema?
- ¿representa una aplicación de N en N ?

Dadas dos máquinas de Turing

- ¿ realizan la misma función?

Para cualquier programa en cualquier computador

- ¿ Se quedará sin memoria antes de terminar la ejecución?
- ¿ Cuanto debemos esperar para estar seguros de que no tiene solución?

El Problema de la parada

sean

n el número código de la maquina de Turing T

m el problema planteado a la máquina

Supongamos que existe H, una máquina que determina si (n,m) termina. Llegaremos a una contradicción:

Construimos K tal que

es composición de K1, K2(≡H), K3

K1 duplica m

K2(≡ H) resuelve si hay parada para (m,m)

K3 actúa inversamente al resultado de K2

Si damos a K su propio número código (k)

K2(≡ H) analiza si K para. Si H no para, no hay solución. Si H termina, no puede dar la solución correcta, ya que en todo caso K se comporta a la inversa.

2 Ejercicios

1)Construir una Máquina de Turing que entre en un ciclo infinito en caso de encontrarse situada frente a un “0” y termine en caso de encontrarse frente a un “1”.

2)Para un alfabeto binario, construir una Máquina de Turing que duplique la entrada.

Soluciones

```
0 1 1 R H;  Esta frente a "1" → termina
0 0 0 R 1;  Esta frente a "0" → nuevo estado para ciclo infinito
1 # # L 0;  estado para ciclo infinito: vuelve a situación inicial
1 0 0 L 0;
1 1 1 L 0;
```

```
0 0 0 L 0;  Busca primer símbolo a la izquierda
0 1 1 L 0;
0 # # R 1;
1 0 x R 2;  Marca símbolo leído y va a copiar
1 1 y R 3;
1 # # R H;              si es blanco se ha terminado
2 0 0 R 2;  Atraviesa el original para copiar un 0
2 1 1 R 2;
2 # # R 4;              pasa a la copia (llevando 0)
3 0 0 R 3;  Atraviesa el original para copiar un 1
3 1 1 R 3;
3 # # R 5;              pasa a la copia (llevando 1)
4 0 0 R 4;  Atraviesa la copia para copiar un 0
4 1 1 R 4;
4 # 0 L 6;              pone el 0 y vuelve
5 0 0 R 5;  Atraviesa la copia para copiar un 1
5 1 1 R 5;
5 # 1 L 6;              pone el 1 y vuelve
6 0 0 L 6;  Vuelve a por otro símbolo restaurando las marcas
6 1 1 L 6;
6 # # L 6;
6 x 0 R 1;
6 y 1 R 1;
```

Ejercicios

1. ¿ Que efectos tendría permitir que una maquina de Turing pudiera tambien mantenerse en la misma posición de la cinta en cada ciclo (sin ir a derecha o izquierda) ?

2. La máquina diseñada para resolver la cuestión 1 ¿parará siempre?

3. Tenemos una zona de la cinta acotada por dos “$” que contiene un número indeterminado de “1”s y espacios en blanco. Diseñe una máquina de Turing para compactar esta zona eliminando los espacios.

Solución al problema 3

```
0 # # L 0; Estado 0 busca extremo izquierdo (se sitúa a la
0 1 1 L 0;                                derecha del $)
0 $ $ R 1;
1 # # R 1;  Estado 1 busca 1 o $ hacia la derecha
1 1 # L 2;
1 $ # L 3;
2 # # L 2;  Estado 2 vuelve a la izquierda a poner un 1
2 1 1 R 4;
2 $ $ R 4;
3 # # L 3;  Estado 3 vuelve a la izquierda a poner un $
3 1 1 R 5;
3 $ $ R 5;
4 # 1 R 1;  Estado 4 pone un 1 y repite la  búsqueda
5 # $ R H; Estado 5 pone un $ y termina
```

BIBLIOGRAFÍA

[Eck 95]
David J. Eck,
"The Most Complex Machine",
A. K. Peters.

Contents

[Dewdney 89]
A. K. Dewdney,
"The New Turing Omnibus: 61 Excursions in Computer Science",
Computer Science Press.

CONTENTS