

Alan Turing: màquines i enigmes

Carles Padró

Conferència inaugural 2015-2016
Facultat de Matemàtiques i Estadística



Dades biogràfiques

- 1912 Neix a Londres
- 1923 Estudia a Sherborne School
- 1931-1934 Estudiant a King's College, Cambridge
- 1935 Fellow a King's College, Cambridge
- 1936 *On Computable Numbers, with an Application to the Entscheidungsproblem*
- 1936-1938 Tesi Doctoral, Princeton University
dirigida per Alonzo Church
Systems of Logic Based on Ordinals

Dades biogràfiques

- 1938-1945 Col·labora amb
Government Code and Cypher School (GC&CS), Bletchley Park
Criptoanàlisi de la màquina Enigma
- 1945 Order of the British Empire
- 1945-1947 National Physical Laboratory (NPL)
col·labora amb el disseny de ACE (Automatic Computing Engine)
Primer disseny d'un ordinador de programa emmagatzemat
- 1948 University of Manchester
Software per a Manchester Mark 1
- 1950 *Computing machinery and intelligence*
- 1952 *The Chemical Basis of Morphogenesis*
- 1954 Mor a Wilmslow, Cheshire

El programa de Hilbert

Fonamentar les matemàtiques
en un sistema finit d'axiomes que sigui
consistent i **complet**

El sistema hauria de ser també **decidable**
és a dir, hi hauria d'haver un **algoritme** per decidir la
validesa de qualsevol proposició
(**Entscheidungsproblem**)



El programa de Hilbert no és possible

Kurt Gödel (1931) demostra que qualsevol sistema que permeti descriure l'aritmètica dels nombres naturals **no pot ser** consistent i complet

Demostra també que la consistència no es pot demostrar dintre del sistema

Alonzo Church i Turing (1936) mostren, independentment, que la **lògica de primer ordre** no és decidable







Entscheidungsproblem

Existeix un **algoritme** per decidir la validesa de qualsevol proposició?

Però, què és un **algoritme**?

Aquesta segona pregunta és especialment important perquè la resposta a la primera és negativa

En les seves solucions de l'Entscheidungsproblem, Church i Turing donen dues respostes diferents

Càlcul lambda i **Màquina de Turing**, respectivament

Els dos models són equivalents **Kleene (1936), Turing (1936)**

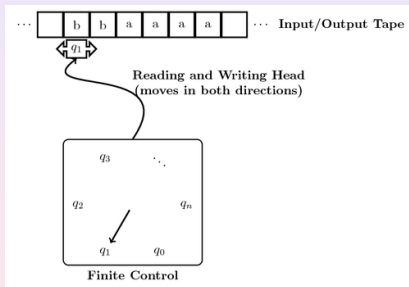
Però, segons Turing, el seu model és
possibly more convincing

Gödel, i el mateix Church, hi estaven d'acord

Tesi de Church-Turing

1936 *On Computable Numbers, with an Application to the Entscheidungsproblem*

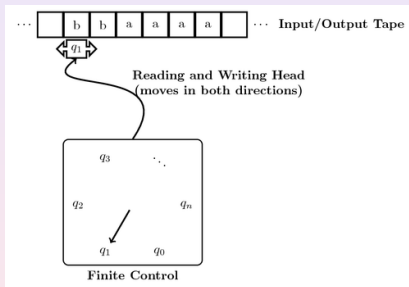
S'hi introdueixen les (universal) computing machines,
que ben aviat s'anomenaran (universal) Turing machines



Tesi de Church-Turing

1936 *On Computable Numbers, with an Application to the Entscheidungsproblem*

S'hi introdueixen les (universal) computing machines, que ben aviat s'anomenaran (universal) Turing machines



Definition (Tesi de Church-Turing)

Qualsevol algoritme efectiu es pot realitzar amb una màquina de Turing

Tesi de Church-Turing

1936 *On Computable Numbers, with an Application to the Entscheidungsproblem*

S'hi introdueixen les (universal) computing machines, que ben aviat s'anomenaran (universal) Turing machines



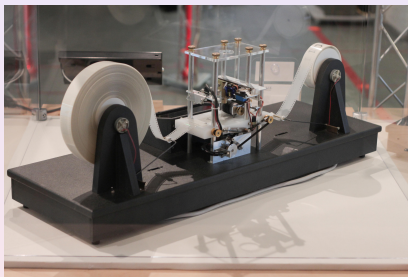
Definition (Tesi de Church-Turing)

Qualsevol algoritme efectiu es pot realitzar amb una màquina de Turing

Tesi de Church-Turing

1936 *On Computable Numbers, with an Application to the Entscheidungsproblem*

S'hi introdueixen les (universal) computing machines, que ben aviat s'anomenaran (universal) Turing machines

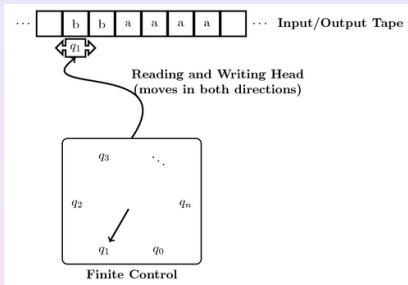


Definition (Tesi de Church-Turing)

Qualsevol algoritme efectiu es pot realitzar amb una màquina de Turing

Màquina de Turing universal

Cada màquina de Turing té una taula d'instruccions o programa **finit**



La **màquina Turing universal** pot fer la tasca de qualsevol màquina de Turing

Només cal codificar la taula d'instruccions en un nombre (**description number, DN**) que posem a la cinta

La màquina de Turing universal va inspirar els **ordinadors de programa emmagatzemat**

On computable numbers

Un nombre real és computable si hi ha una màquina de Turing que pot escriure la seva representació decimal (o binària) a la cinta

Com que cada màquina de Turing es pot descriure amb un nombre natural (DN), hi ha només una quantitat numerable de nombres computables

Els nombres algebraics són computables

El límit d'una successió **computablement convergent** de nombres computables és computable

Per exemple

$$\pi = 4 \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots \right) \quad e = 1 + 1 + \frac{1}{2!} + \frac{1}{3!} + \cdots$$

Problemes indecidibles

No hi ha cap màquina de Turing D per decidir si un nombre donat és el DN d'una màquina de Turing que genera una seqüència infinita

Altrament, podem dissenyar a partir de D una màquina de Turing H que genera la seqüència β'

DN_1	1	0	1	1	0	0	0	0	1	...
DN_2	0	0	1	0	0	1	0	1	1	...
DN_3	1	1	0	1	0	0	0	1	0	...
DN_4	1	1	1	1	0	1	0	0	1	...
DN_5	1	0	0	1	0	0	1	0	1	...
DN_6	0	1	1	1	0	0	0	1	0	...
DN_7	0	1	1	0	0	0	0	0	1	...
DN_8	1	1	1	1	1	0	1	0	0	...
DN_9	1	0	0	1	0	0	0	0	1	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Problemes indecidibles

No hi ha cap màquina de Turing D per decidir si un nombre donat és el DN d'una màquina de Turing que genera una seqüència infinita

Altrament, podem dissenyar a partir de D una màquina de Turing H que genera la seqüència β'

DN_1	1	0	1	1	0	0	0	0	1	...
DN_2	0	0	1	0	0	1	0	1	1	...
DN_3	1	1	0	1	0	0	0	1	0	...
DN_4	1	1	1	1	0	1	0	0	1	...
DN_5	1	0	0	1	0	0	1	0	1	...
DN_6	0	1	1	1	0	0	0	1	0	...
DN_7	0	1	1	0	0	0	0	0	1	...
DN_8	1	1	1	1	1	0	1	0	0	...
DN_9	1	0	0	1	0	0	0	0	1	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

En algun moment, H troba el **seu propi DN**, i cau en un bucle
 $\implies$ contradicció

Problemes indecidibles

No hi ha cap màquina de Turing D per decidir si un nombre donat és el DN d'una màquina de Turing que genera una seqüència infinita

Altrament, podem dissenyar a partir de D una màquina de Turing H que genera la seqüència β'

En algun moment, H troba el **seu propi DN**, i cau en un bucle
 $\implies$ contradicció

A partir d'aquí es troben altres problemes indecidibles

- Decidir si una màquina donada imprimirà 0 alguna vegada
- Decidir si una màquina donada s'aturarà

Finalment, això permet resoldre l'Entscheidungsproblem

El pare de la informàtica teòrica

Es considera que Turing és el pare de la informàtica teòrica (*theoretical computer science*) sobretot per la introducció de la màquina de Turing com a model de computació

Altres idees seminals de Turing

- Subrutines (*skeleton tables*)
- Oracles
- Ordinadors de programa emmagatzemat

En particular, la **teoria de la complexitat** sorgeix naturalment de les contribucions de Turing

Què podem **realment** calcular?

Quins algoritmes són eficients?

Es considera que els algoritmes eficients són els que funcionen en **temps polinòmic**. . . però, per què?

Extensió de la tesi de Church-Turing

Qualsevol parell de models generals i raonables de computació seqüencial estan relacionats polinòmicament

Tesi de Cobham-Edmonds (1965)

És factible resoldre un problema computacional només si es pot fer en temps polinòmic

Aquesta classe d'algoritmes és tancada per composició

Classes de complexitat P, NP

La **classe de complexitat P** està formada pels problemes que es poden resoldre en temps polinòmic

Per exemple, programació lineal, primalitat, ...

La **classe de complexitat NP** està formada pels problemes de **decisió** tals que, si la resposta és afirmativa, això es pot demostrar en temps polinòmic

Òbviament, $P \subset NP$

NP vol dir **no determinista en temps polinòmic**

Un problema és **NP-complet**
si qualsevol altre problema NP s'hi pot reduir
Són els més difícils de la classe NP

Per exemple, decidir si un graf té un cicle de Hamilton

Provar que $P \neq NP$ és un dels principals problemes oberts en matemàtiques

La utilitat dels problemes difícils

Criptografia de clau privada

Emisor i receptor comparteixen una **clau secreta** k

$$c = E(k, m) \quad m = D(k, c)$$

Algoritmes eficients per a E, D

És **difícil** determinar m a partir de c si no es coneix k

És **difícil** determinar k a partir de m i c

Acordar la clau secreta és el principal obstacle

La utilitat dels problemes difícils

Criptografia de clau pública

El receptor té una **clau pública** pk i una **clau privada** sk

$$c = E(pk, m) \quad m = D(sk, c)$$

No cal acordar una clau

Algoritmes eficients per a E , D

Ha de ser difícil trobar m a partir de c, pk si no es coneix sk

També ha de ser difícil trobar sk a partir de pk , etc

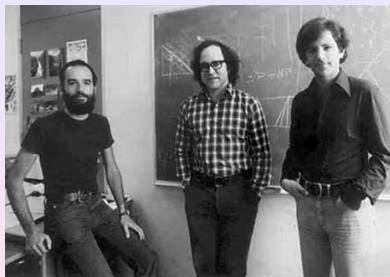
Amb la mateixa idea, obtenim **signatures digitals** i protocols per **acordar una clau**

Diffie-Hellman, 1976

Rivest-Shamir-Adleman, 1977



Ralph Merkle, Martin Hellman,
Whitfield Diffie



Adi Shamir, Ron Rivest,
Leonard Adleman

La utilitat dels problemes difícils

Com poden Alice i Bob acordar una clau?

- Alice i Bob acorden, públicament, un grup cíclic G i un generador g (subgrup del grup multiplicatiu d'un cos primer o del grup de punts d'una corba el·líptica)
- Alice selecciona x aleatori i envia g^x a Bob
- Bob selecciona y aleatori i envia g^y a Alice
- Alice i Bob han acordat la clau $g^{xy} = (g^x)^y = (g^y)^x$

És **difícil** distingir entre

$$(g, g^x, g^y, g^{xy}) \quad (g, g^x, g^y, g^z)$$

Diffie-Hellman, 1976

Hi ha molts problemes difícils, però només alguns es poden aplicar a la criptografia de clau pública

En una carta a la família des de Princeton, Turing escriu:

I have just discovered a possible application of the kind of thing I am working on at present. It answers the question 'What is the most general kind of code or cipher possible', and at the same time (rather naturally) enables one to construct a lot of particular and interesting codes. One of them is pretty well impossible to decode without the key and very quick to encode. I expect I could sell them to H.M. Government for quite a substantial sum, but am rather doubtful about the morality of such things. . .

Alan Turing, criptoanalista

Poc després de tornar de Princeton, Turing comença a col·laborar amb **Government Code and Cypher School (GC&CS)**

Al setembre de 1939, quan comença la Segona Guerra Mundial, Turing s'instal·la a **Bletchley Park**, nova seu de la GC&CS



Alan Turing, criptoanalista

Poc després de tornar de Princeton, Turing comença a col·laborar amb **Government Code and Cypher School (GC&CS)**

Al setembre de 1939, quan comença la Segona Guerra Mundial, Turing s'instal·la a **Bletchley Park**, nova seu de la GC&CS



Alan Turing, criptoanalista

Poc després de tornar de Princeton, Turing comença a col·laborar amb **Government Code and Cypher School (GC&CS)**

Al setembre de 1939, quan comença la Segona Guerra Mundial, Turing s'instal·la a **Bletchley Park**, nova seu de la GC&CS



Va preparar una *Emergency list of men of the Professor type*

At certain universities. . . there were men now in senior positions who had worked in our ranks during 1914-18. These men knew the type required. Thus it fell out that our most successful recruiting occurred from these universities. During 1937 and 1938 we were able to arrange a series of courses to which we invited our recruits to give them even a dim idea of what would be required fo them. . . These men joined up in September 1939.

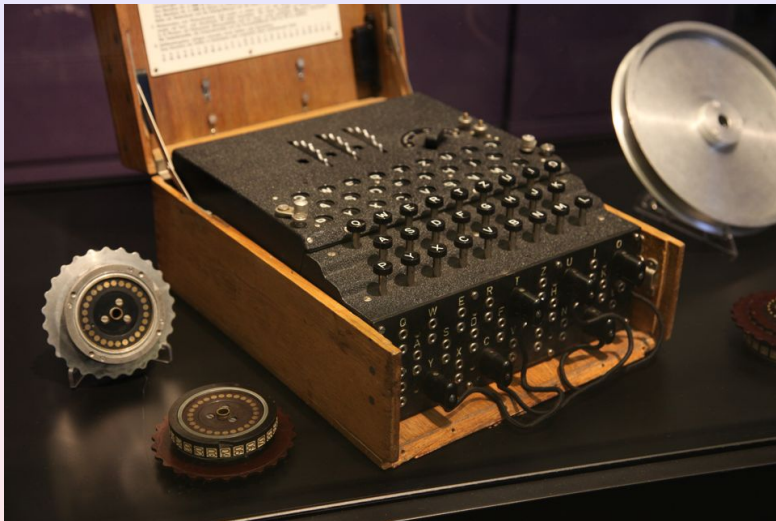


Al setembre de 1939, es van ajuntar unes 30 persones a Bletchley Park, moltes, com Turing, de la *emergency list*

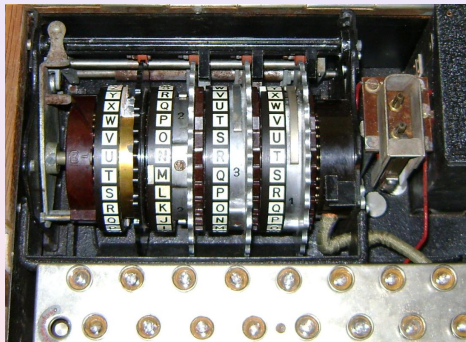
Amb poc temps, Bletchley Park es va convertir en una fàbrica de trencar xifrats

Cap al final de la guerra, unes 9.000 persones hi treballaven

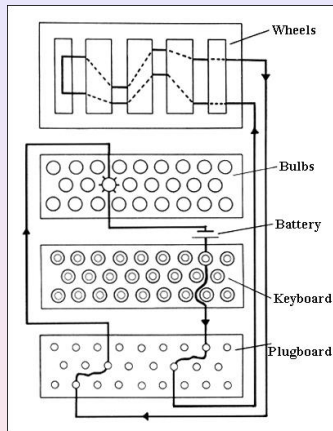
La màquina Enigma



La màquina Enigma



La màquina Enigma



És un sistema de xifrat de **clau privada**

Els procediments de xifrat i desxifrat són **simètrics**

Criptoanàlisi de la màquina Enigma

Dilly Knox va trencar la **màquina Enigma** usada pels italians a la Guerra Civil Espanyola

Aquesta màquina no tenia panell de connexions.

Els britànics no van poder trencar encara la màquina Enigma usada per l'exercit alemany

Els matemàtics polonesos

Marian Rejewski, Henryk Zygalski i Jerzy Różycki,
van fer grans avenços al llarg dels anys 30

En particular, van dissenyar i construir la **bomba kryptologiczna**,
la primera màquina per trencar l'Enigma

Al juliol de 1939, els polonesos comuniquen els seus avenços als francesos i als britànics

Criptoanàlisi de la màquina Enigma

A partir d'aquí, la major part de la criptoanàlisi es va dur a terme a Bletchley Park

Una de les vulnerabilitats de la màquina Enigma es devia a que una lletra mai se substituïa per ella mateixa

Però un missatge xifrat ha de semblar aleatori!

Aquesta restricció disminueix l'aleatorietat i facilita l'atac amb missatge en clar conegut

Per exemple, considerem el sistema de xifrat **one-time pad**

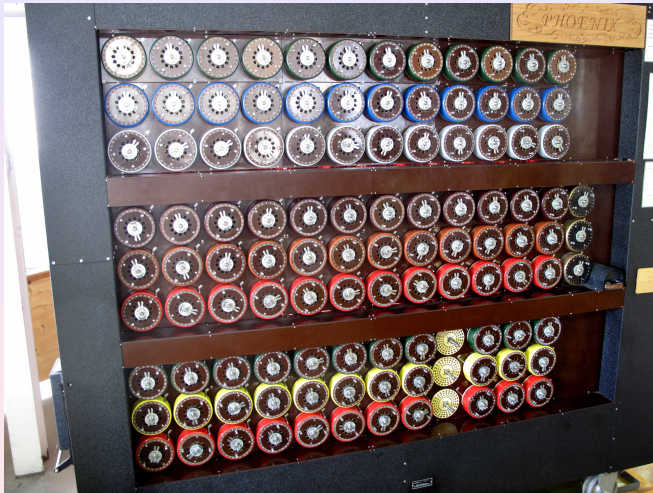
Missatge 11001000110101000101

Clau **00101111010111011100**

Xifrat 11100111100010011001

Altres factors que debilitaven la seguretat eren restriccions en l'aleatorietat de les claus, missatges massa predictibles, operadors maldestres o mandrosos, etc

La màquina Bombe



La màquina Bombe

La criptoanàlisi de la màquina Enigma es realitzava amb la **Bombe**, una màquina dissenyada per Turing

S'usava per determinar l'ordre dels rotors, la posició inicial dels rotors i les connexions del panell

Moltes especificacions es podien descartar ràpidament per contradicció

Quan la màquina trobava una configuració possible, s'aturava

GCHQ, el successor de GC&CS



Signatura digital

La clau privada s'usa per a **signar**, la clau pública per a **verificar**

Signatura: $s = D(sk, m)$

Verificació: donat el parell (m, s) , es comprova si $m = E(pk, s)$

Els missatges m solen ser massa llargs, el que fa les operacions massa costoses

Per tant, se signa un **resum** $H(m)$ del missatge, on H és una funció de resum (**hash function**) acordada

Signatura: $s = D(sk, H(m))$

Verificació: donat el parell (m, s) , es comprova si $H(m) = E(pk, s)$

Signatura: $s = D(sk, H(m))$

Verificació: donat el parell (m, s) , es comprova si $H(m) = E(pk, s)$

Si $H(m) = H(m')$, els dos parells missatge-signatura (m, s) , (m', s) són vàlids

Hi ha moltes **colisions** $H(m) = H(m')$

La funció resum és **segura** si és difícil trobar-les

La majoria de colisions són inofensives perquè m' no té sentit

L'any 2009, **Marc Stevens** i altres van presentar un mètode per trobar colisions **interessants** per a la funció resum MD5, i van mostrar com usar-lo per obtenir **certificats** falsos

L'any 1994, **Peter Shor** va presentar un algorisme de factorització que funciona amb temps polinòmic en un **ordinador quàntic**

L'algorisme es pot adaptar al **logaritme discret**

Per tant, els sistemes criptogràfics de clau pública usats actualment seran insegurs quan es construeixi un ordinador quàntic

S'està treballant ja en la **criptografia post-quàntica**, basada en problemes difícils com la correcció d'errors en codis lineals i problemes en reticles