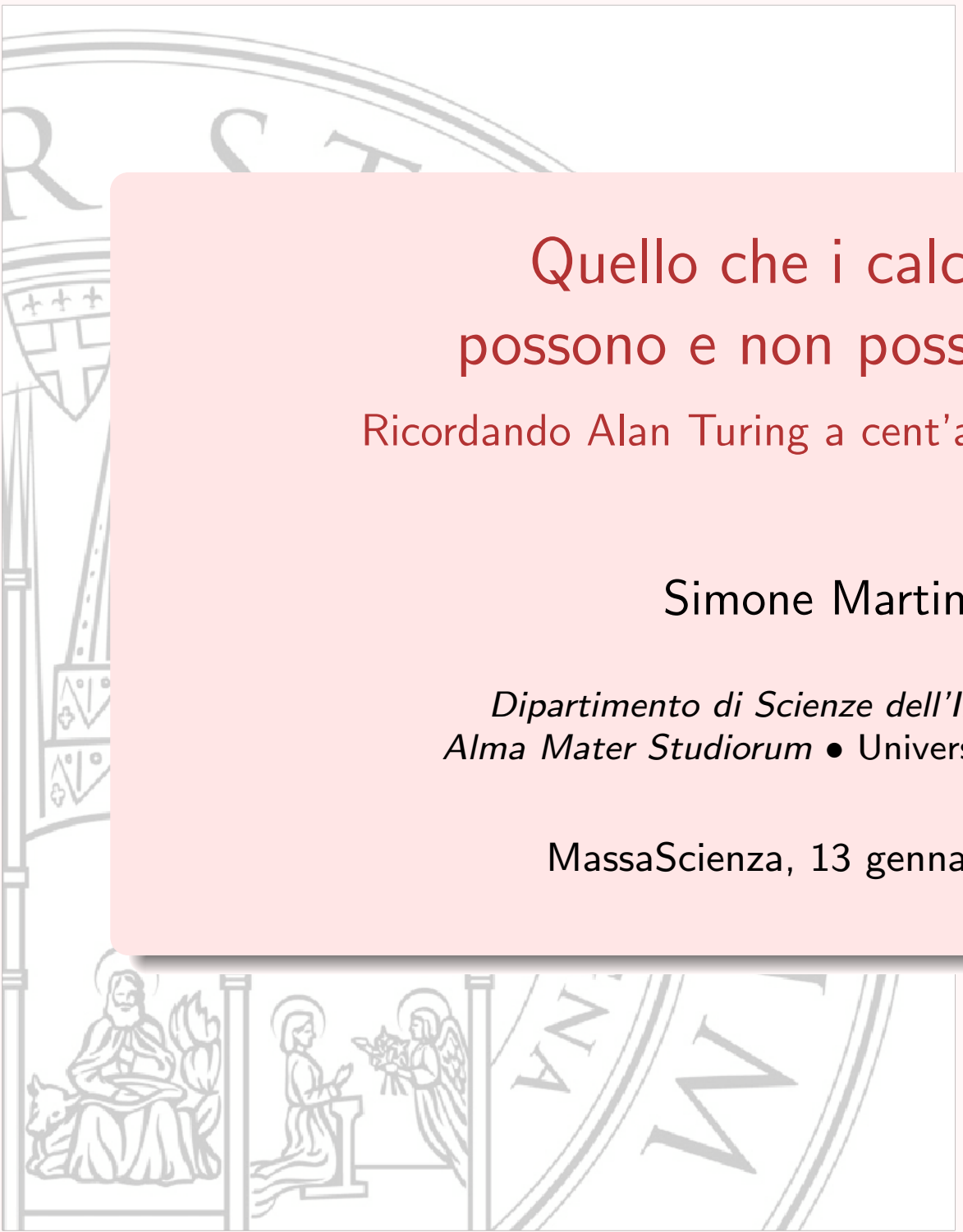


Quello che i calcolatori possono e non possono fare

Ricordando Alan Turing a cent'anni dalla nascita

Simone Martini

Dipartimento di Scienze dell'Informazione
Alma Mater Studiorum • Università di Bologna

MassaScienza, 13 gennaio 2012

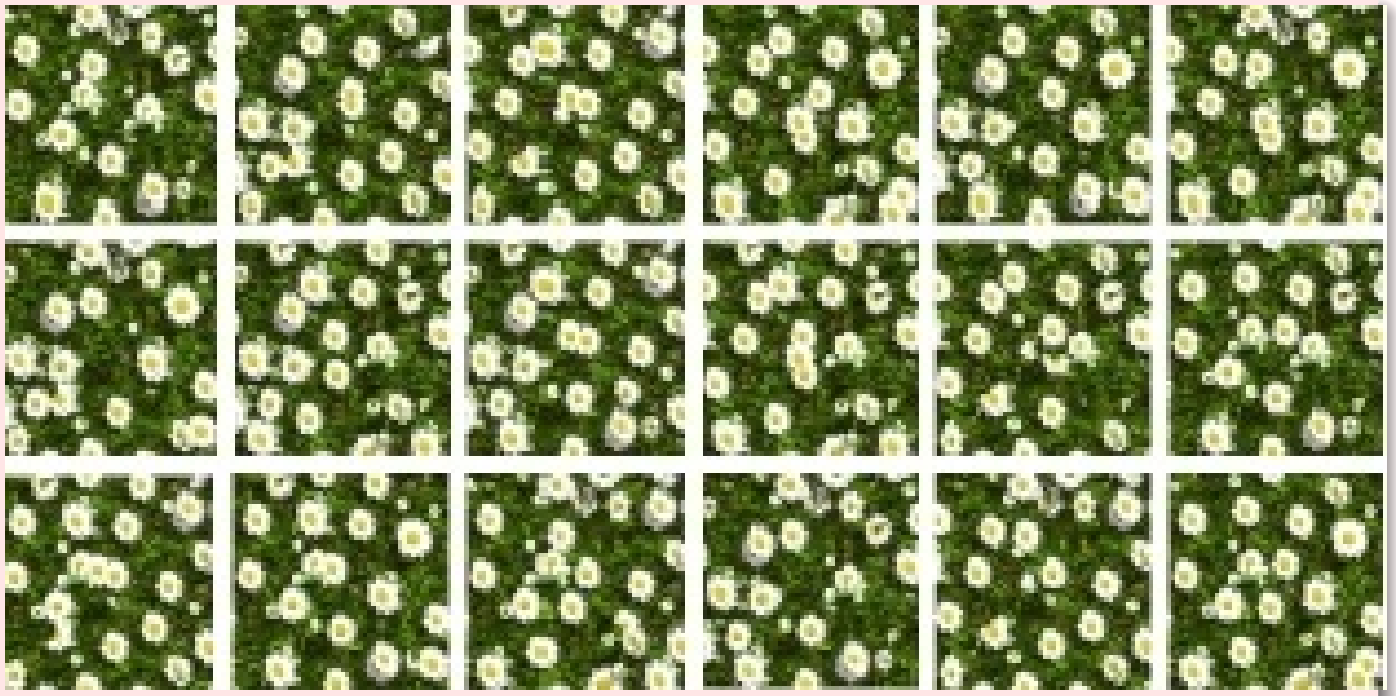
Indice

Indice

Un bel prato di margherite...

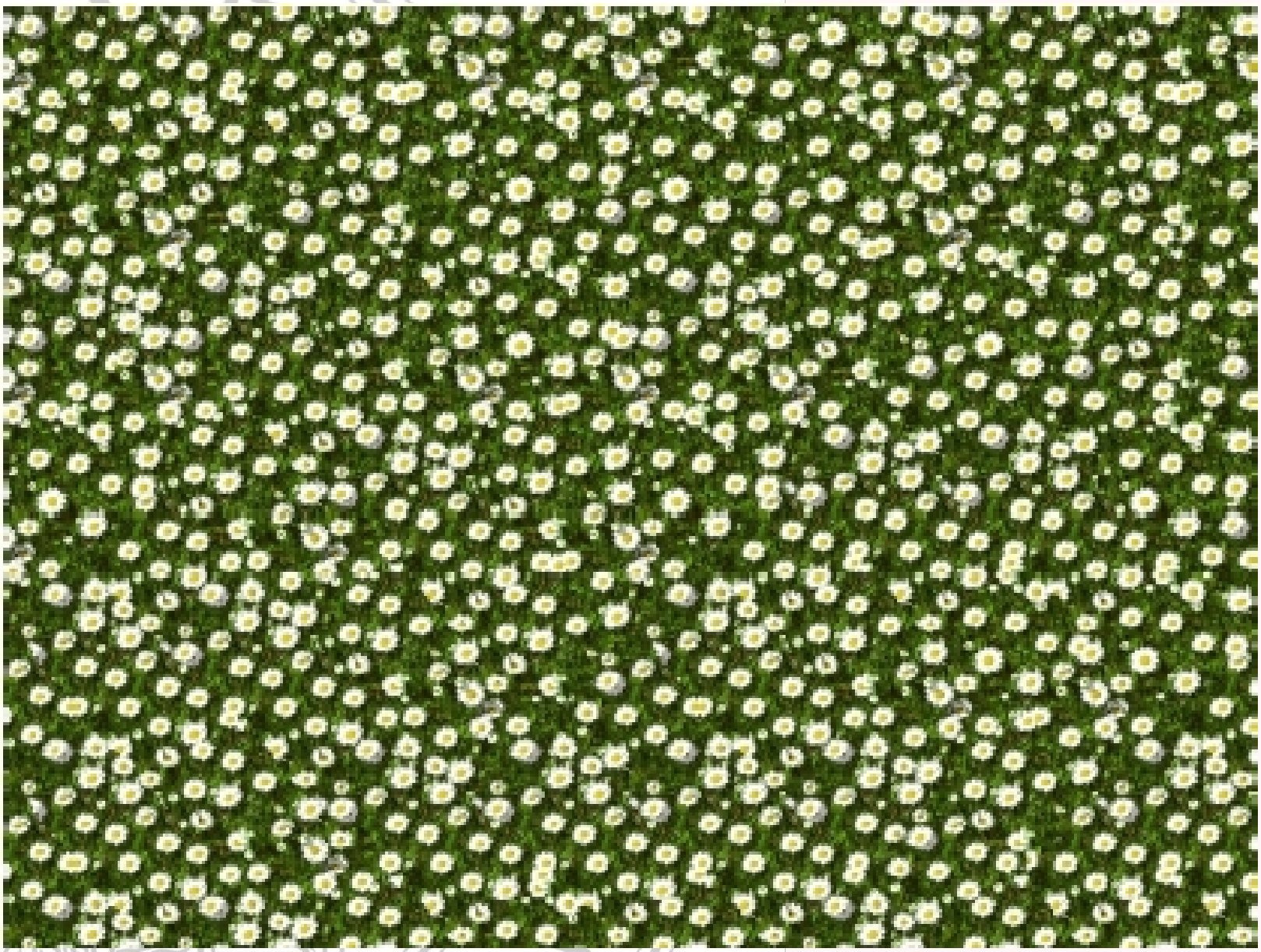
. . . semi-sintetiche!

Input

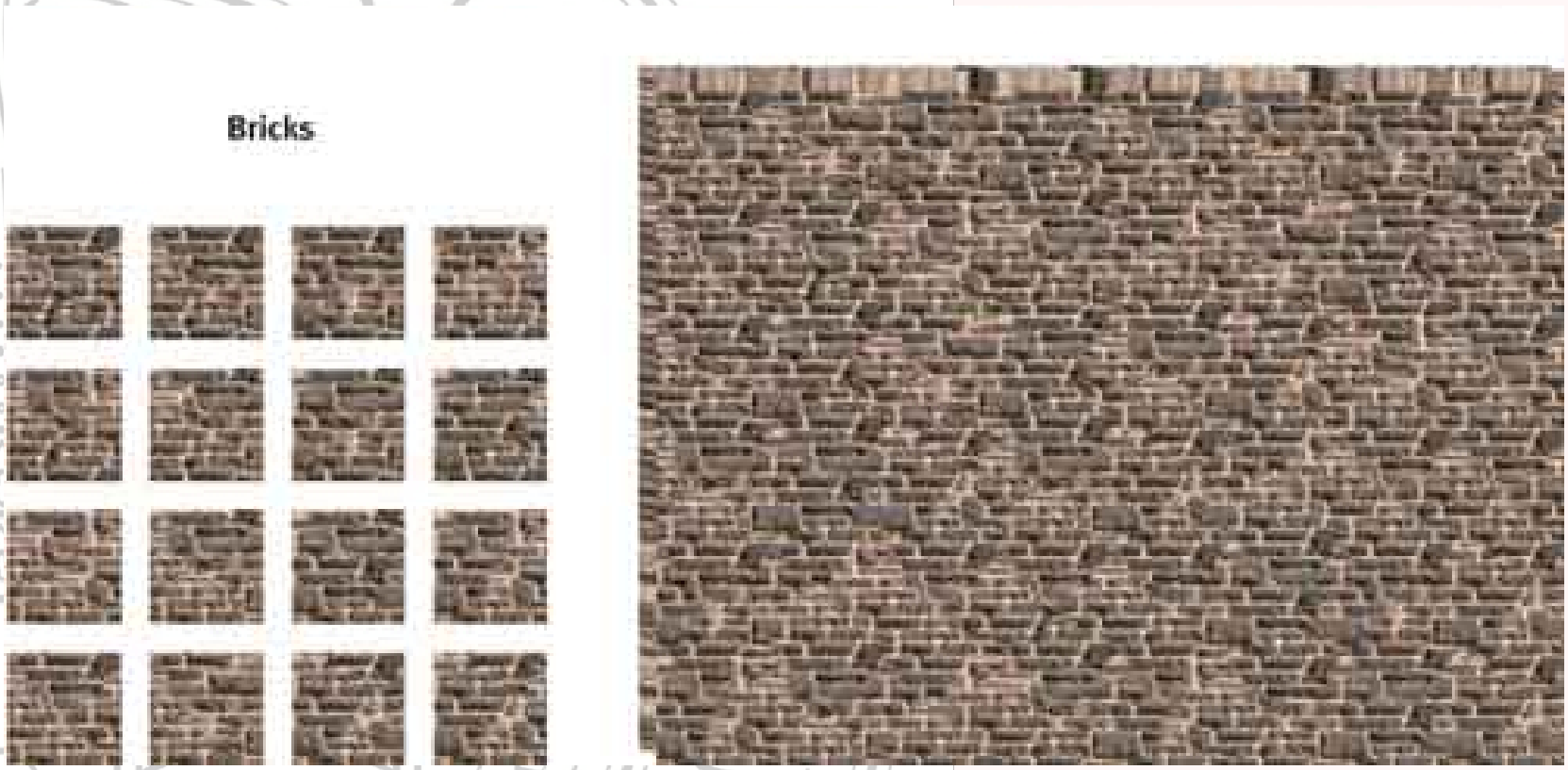


Piastrelle sintetizzate a partire dall'input

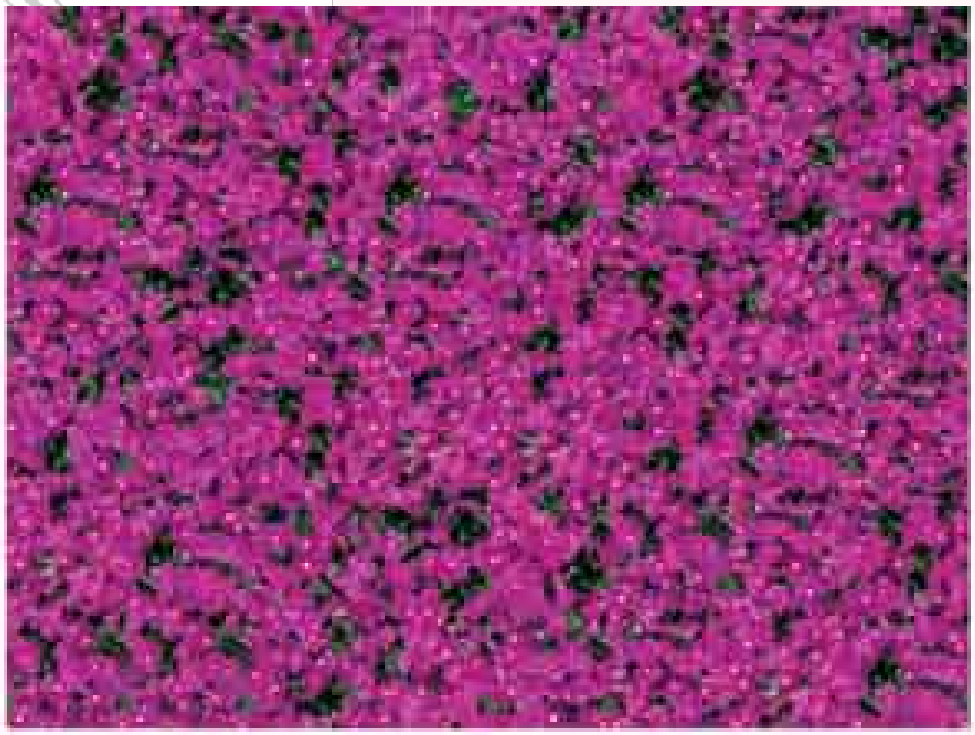
Un prato *non periodico* di margherite



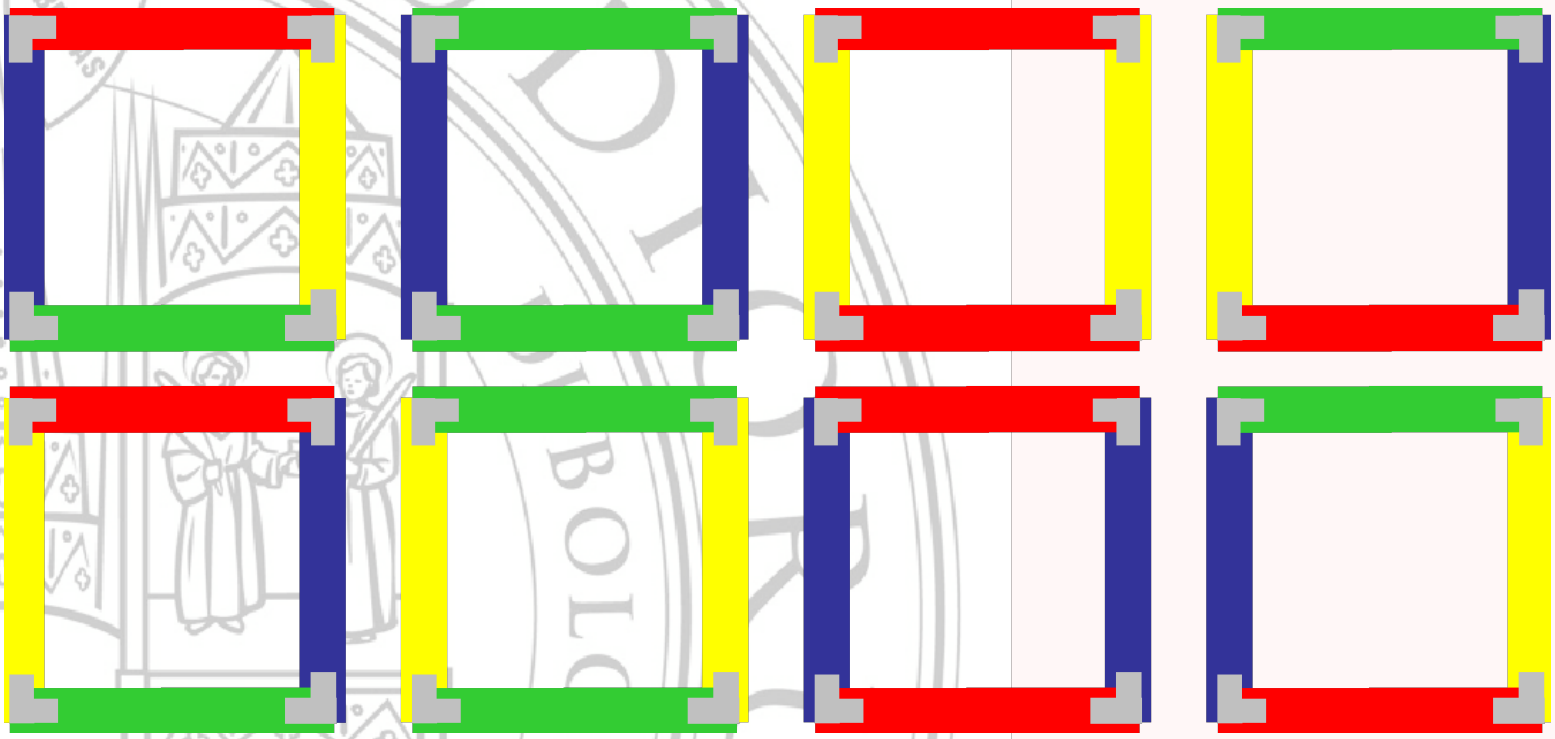
Un muro *non periodico* di mattoni



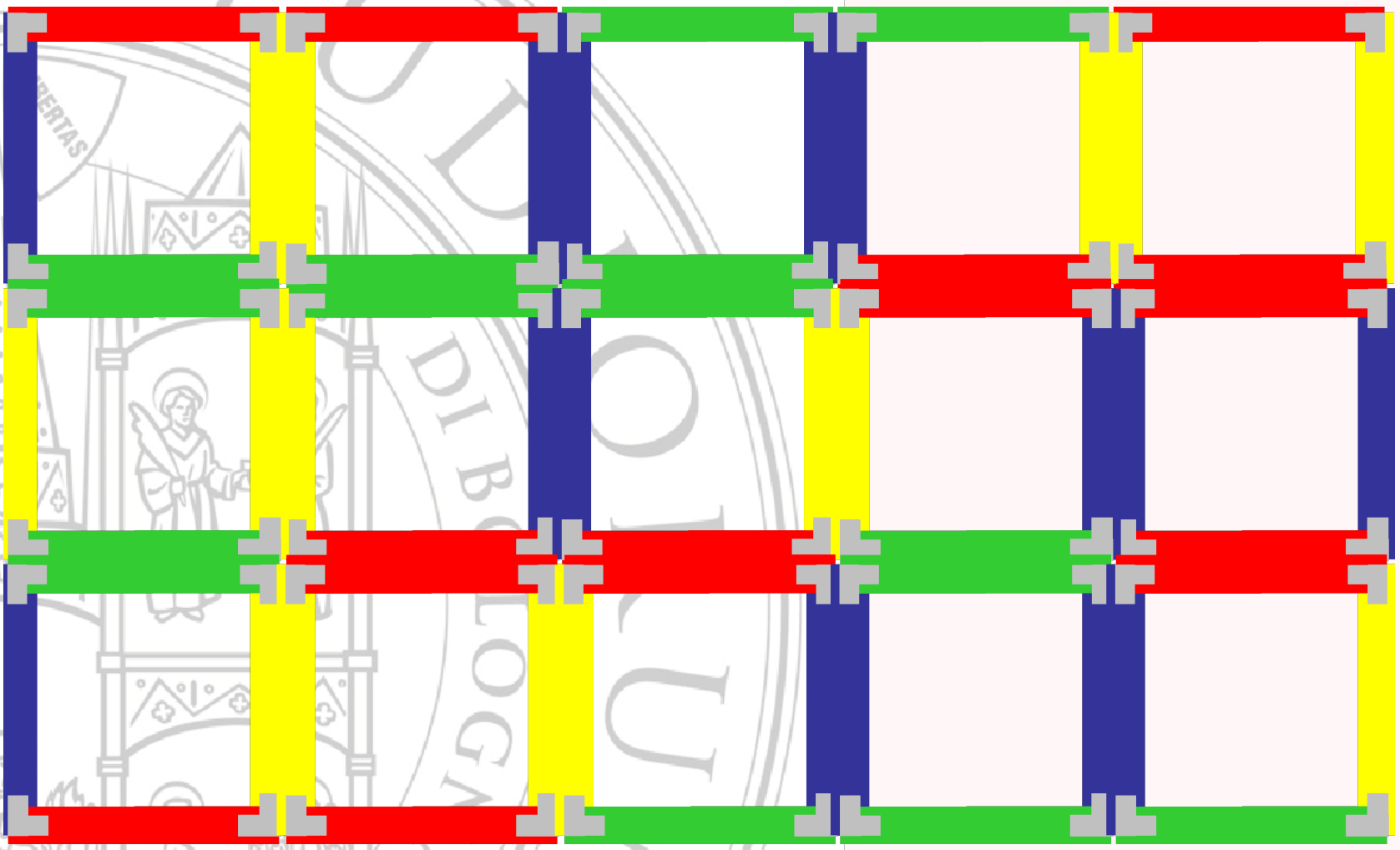
Un altro prato *non periodico*



Piastrelle di Wang

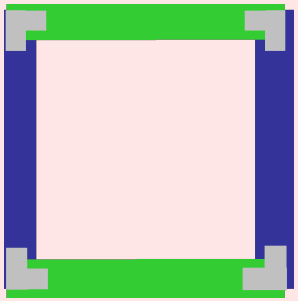


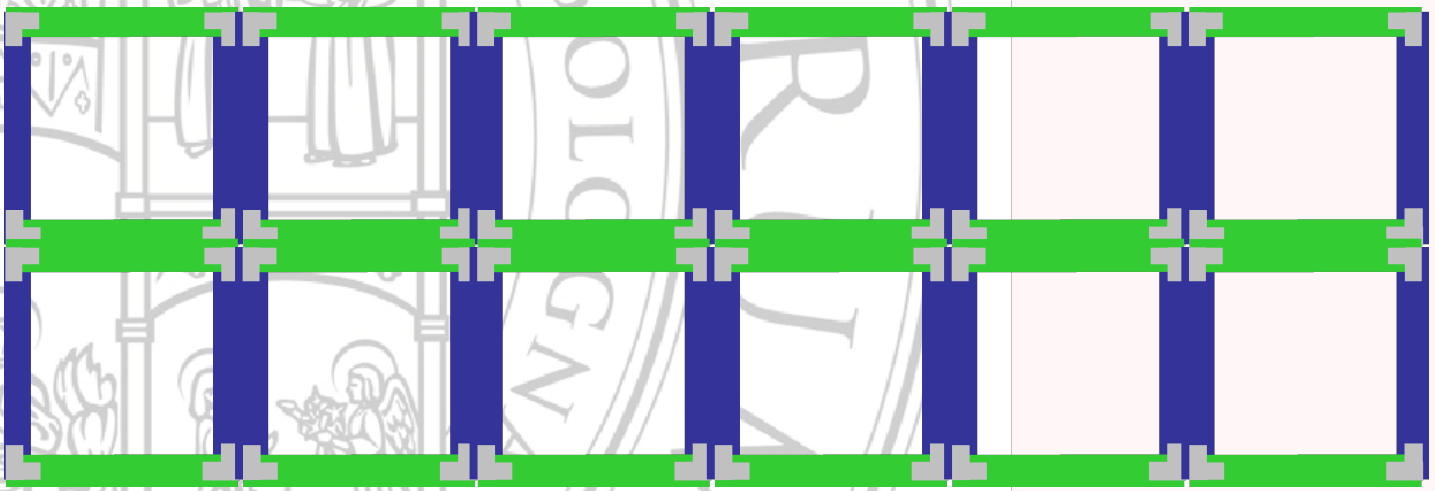
Piastrellatura (*tiling*) di Wang



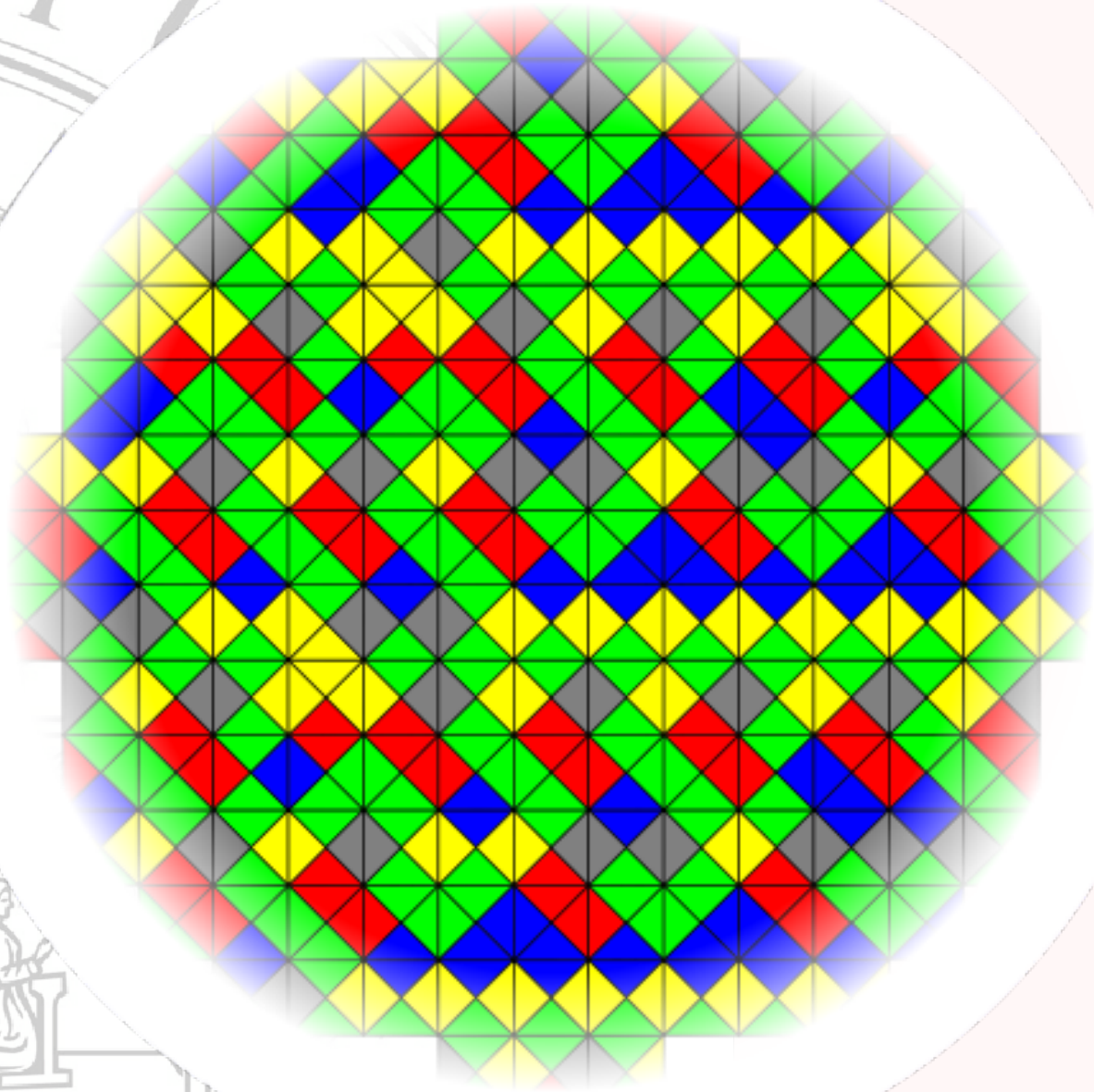
Una piastrellatura semplice periodica

Un solo tipo di piastrella

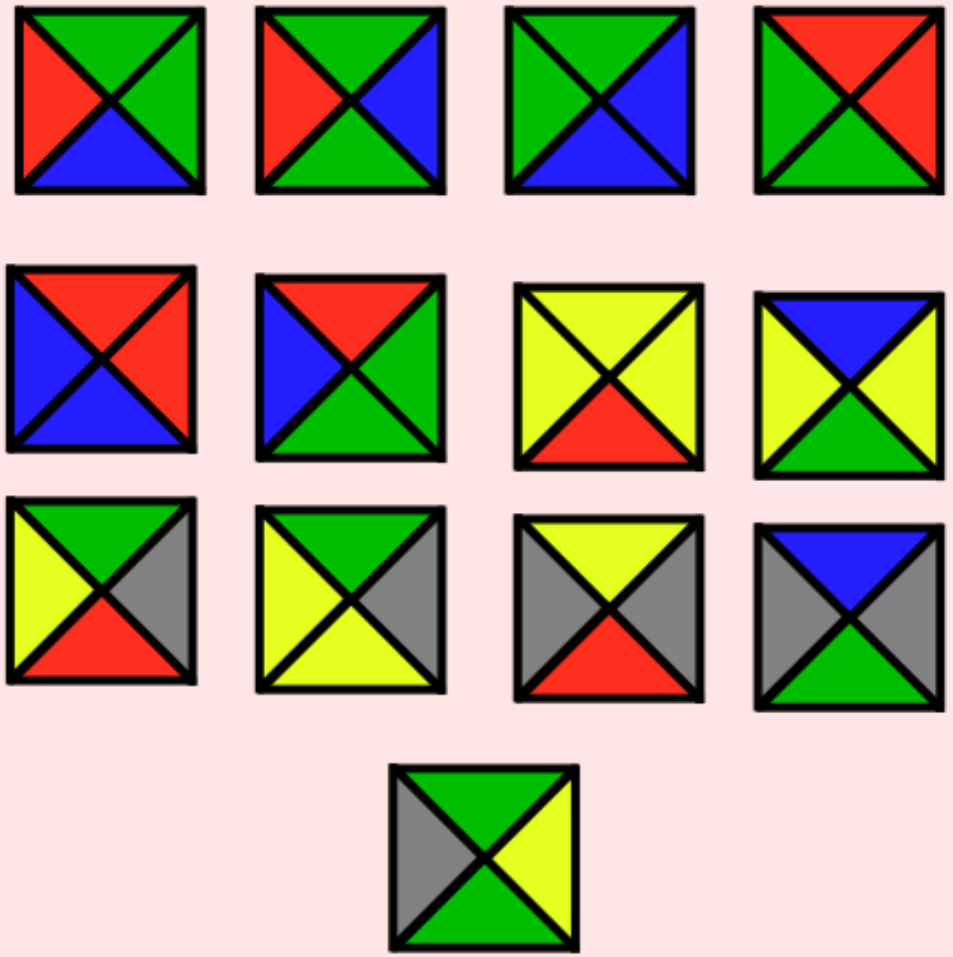




Piastrellature non periodiche



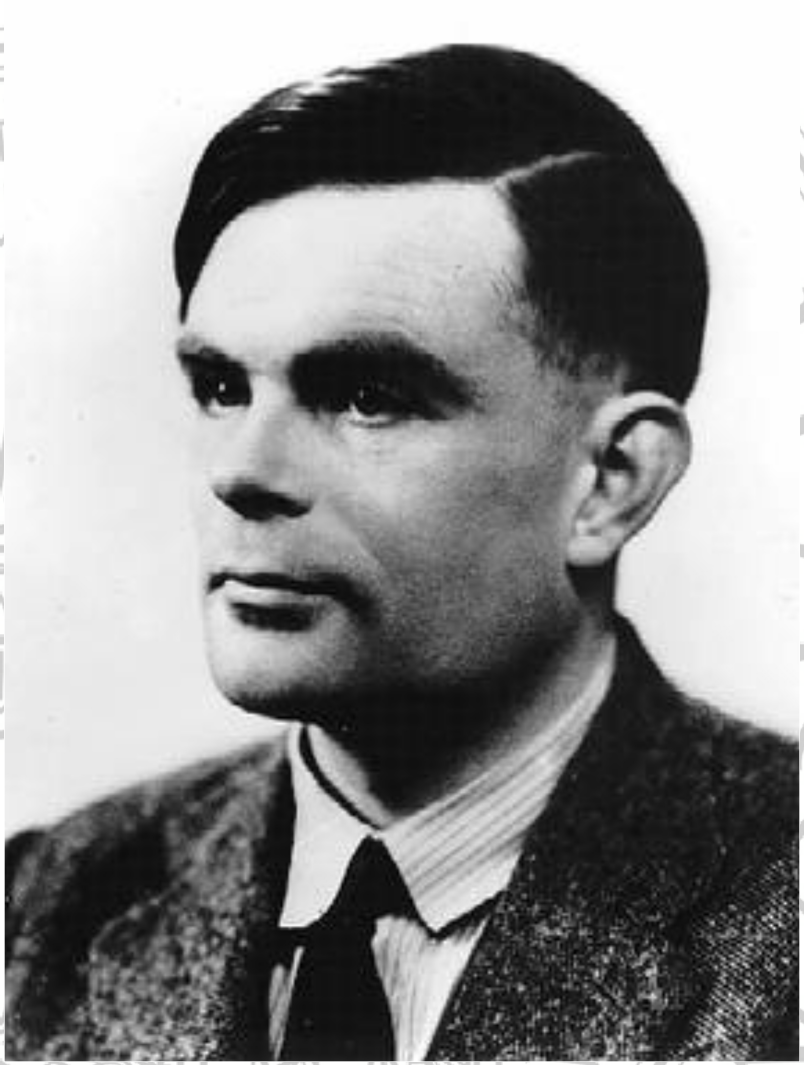
E le relative piastrelle



Permettono solo piastrellature non periodiche.

Indice

Alan M. Turing (1912–1954)



nato: 23 giugno 1912

OBE, *Order of the British Empire*
FRS, *Fellow of the Royal Society*

Criptoanalista
Matematico
Logico

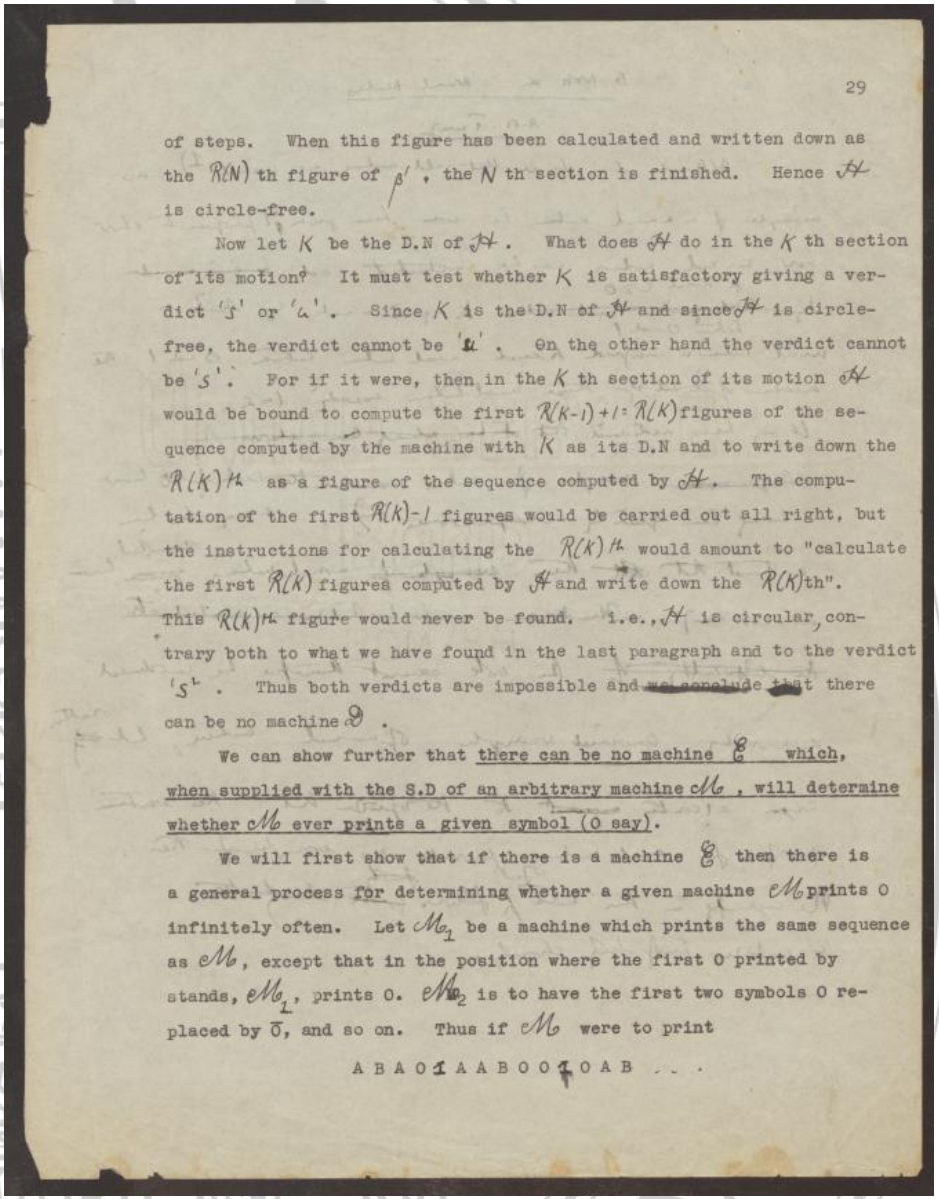
Alan M. Turing (1912–1954)

nato: 23 giugno 1912

OBE, *Order of the British Empire*
FRS, *Fellow of the Royal Society*

Criptoanalista
Matematico
Logico

1936: Cosa significa calcolare?



29

of steps. When this figure has been calculated and written down as the $R(N)$ th figure of β', the N th section is finished. Hence $\mathcal{H}$ is circle-free.

Now let K be the D.N of $\mathcal{H}$. What does $\mathcal{H}$ do in the K th section of its motion? It must test whether K is satisfactory giving a verdict 's' or 'u'. Since K is the D.N of $\mathcal{H}$ and since $\mathcal{H}$ is circle-free, the verdict cannot be 'u'. On the other hand the verdict cannot be 's'. For if it were, then in the K th section of its motion $\mathcal{H}$ would be bound to compute the first $R(K-1)+1 = R(K)$ figures of the sequence computed by the machine with K as its D.N and to write down the $R(K)$ th as a figure of the sequence computed by $\mathcal{H}$. The computation of the first $R(K)-1$ figures would be carried out all right, but the instructions for calculating the $R(K)$ th would amount to "calculate the first $R(K)$ figures computed by $\mathcal{H}$ and write down the $R(K)$th". This $R(K)$th figure would never be found. i.e., $\mathcal{H}$ is circular, contrary both to what we have found in the last paragraph and to the verdict 's'. Thus both verdicts are impossible and we conclude that there can be no machine $\mathcal{D}$.

We can show further that there can be no machine $\mathcal{E}$ which, when supplied with the S.D of an arbitrary machine $\mathcal{M}$, will determine whether $\mathcal{M}$ ever prints a given symbol (0 say).

We will first show that if there is a machine $\mathcal{E}$ then there is a general process for determining whether a given machine $\mathcal{M}$ prints 0 infinitely often. Let $\mathcal{M}_1$ be a machine which prints the same sequence as $\mathcal{M}$, except that in the position where the first 0 printed by stands, $\mathcal{M}_1$, prints 0. $\mathcal{M}_2$ is to have the first two symbols 0 replaced by $\overline{0}$, and so on. Thus if $\mathcal{M}$ were to print

A B A 0 1 A A B 0 0 1 0 A B . . .

© P.N. Furbank, Turing Digital Archive

On Computable Numbers, with an Application to the Entscheidungsproblem, Proc. Lond. Math. Soc. (2) 42 pp. 230-265 (1936)

Ben prima delle macchine che calcolano. . .

- Primo calcolatore elettronico:
  ENIAC, 1946; University of Pennsylvania's Moore School of Electrical Engineering

Ma qual è il problema?

Calcolo e matematica

- Fino al settecento non c'è grande differenza:

  si cercano soluzioni
  mediante costruzioni

- Euclide: costruzione con riga e compasso:

Sopra una data retta terminata (segmento) costruire un triangolo equilatero

- Algebra: formula risolutiva dell'equazione di grado n
- Analisi: determinare l'area sottesa ad una curva

Calcolo e matematica: limitazioni

- Alcune costruzioni sono possibili
- Altre sono dimostrabilmente impossibili
- Con riga e compasso:

  trisecazione di un angolo arbitrario (Wantzel, 1837)
  duplicazione del cubo (Wantzel, 1837)
  quadratura del cerchio (von Lindemann, 1882)
- Soluzione "per radicali":

  Formula risolutiva per equazioni di grado ≥ 5 (Ruffini, 1799)
- Area sottesa ad una curva:

  Per alcune curve non esiste una formulazione "chiusa" per l'area sottesa ad essa

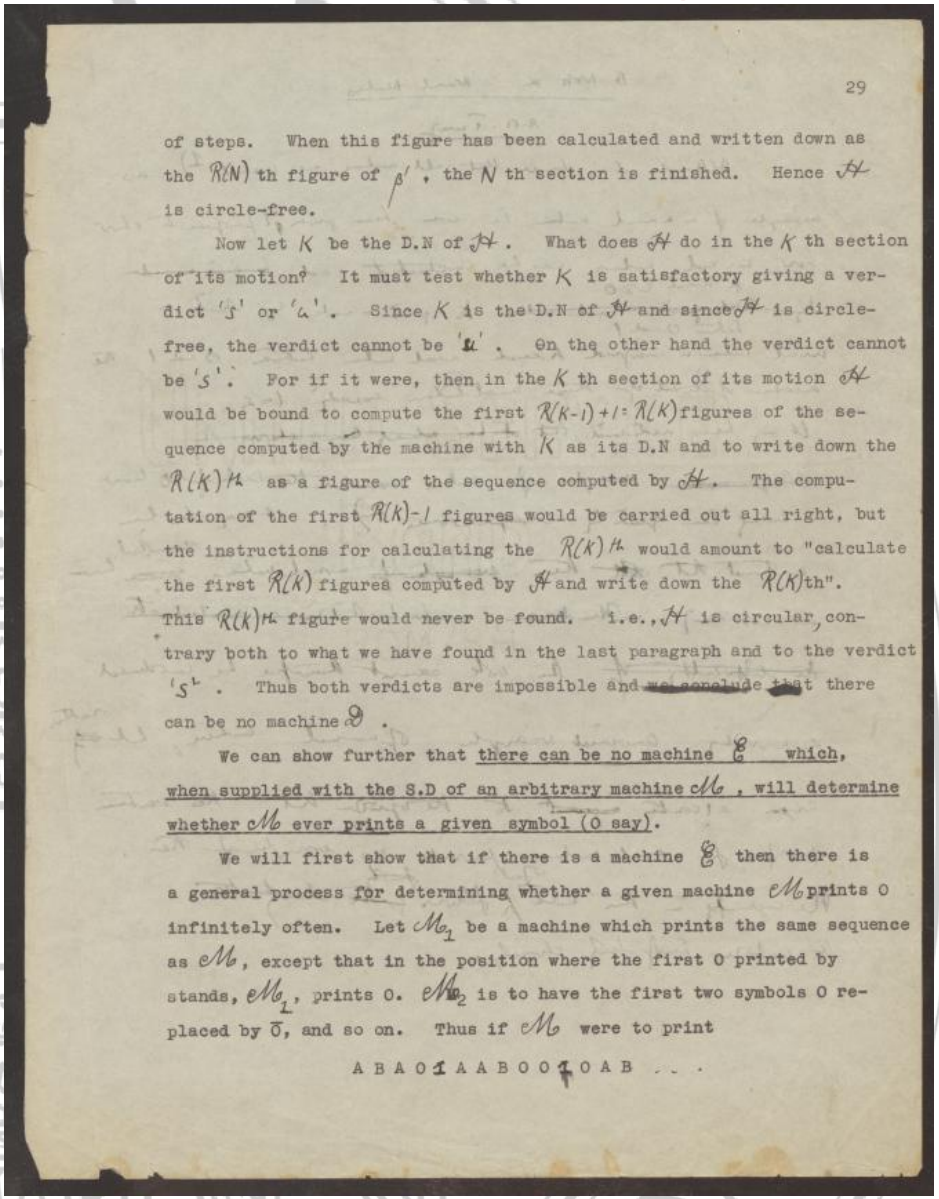
Negatives are such difficult things to prove.

[A. Christie, Three blind mice.]

Matematica e infinito

- Oggetti infiniti (e.g., *un* numero reale)
- Come sono manipolati?
- Descrizioni finite

Turing: Cosa significa calcolare?



29

of steps. When this figure has been calculated and written down as the $R(N)$ th figure of β', the N th section is finished. Hence $\mathcal{H}$ is circle-free.

Now let K be the D.N of $\mathcal{H}$. What does $\mathcal{H}$ do in the K th section of its motion? It must test whether K is satisfactory giving a verdict 's' or 'u'. Since K is the D.N of $\mathcal{H}$ and since $\mathcal{H}$ is circle-free, the verdict cannot be 'u'. On the other hand the verdict cannot be 's'. For if it were, then in the K th section of its motion $\mathcal{H}$ would be bound to compute the first $R(K-1)+1 = R(K)$ figures of the sequence computed by the machine with K as its D.N and to write down the $R(K)$th as a figure of the sequence computed by $\mathcal{H}$. The computation of the first $R(K)-1$ figures would be carried out all right, but the instructions for calculating the $R(K)$th would amount to "calculate the first $R(K)$ figures computed by $\mathcal{H}$ and write down the $R(K)$th". This $R(K)$th figure would never be found. i.e., $\mathcal{H}$ is circular, contrary both to what we have found in the last paragraph and to the verdict 's'. Thus both verdicts are impossible and we conclude that there can be no machine $\mathcal{D}$.

We can show further that there can be no machine $\mathcal{E}$ which, when supplied with the S.D of an arbitrary machine $\mathcal{M}$, will determine whether $\mathcal{M}$ ever prints a given symbol (0 say).

We will first show that if there is a machine $\mathcal{E}$ then there is a general process for determining whether a given machine $\mathcal{M}$ prints 0 infinitely often. Let $\mathcal{M}_1$ be a machine which prints the same sequence as $\mathcal{M}$, except that in the position where the first 0 printed by stands, $\mathcal{M}_1$, prints 0. $\mathcal{M}_2$ is to have the first two symbols 0 replaced by $\overline{0}$, and so on. Thus if $\mathcal{M}$ were to print

A B A 0 1 A A B 0 0 1 0 A B . . .

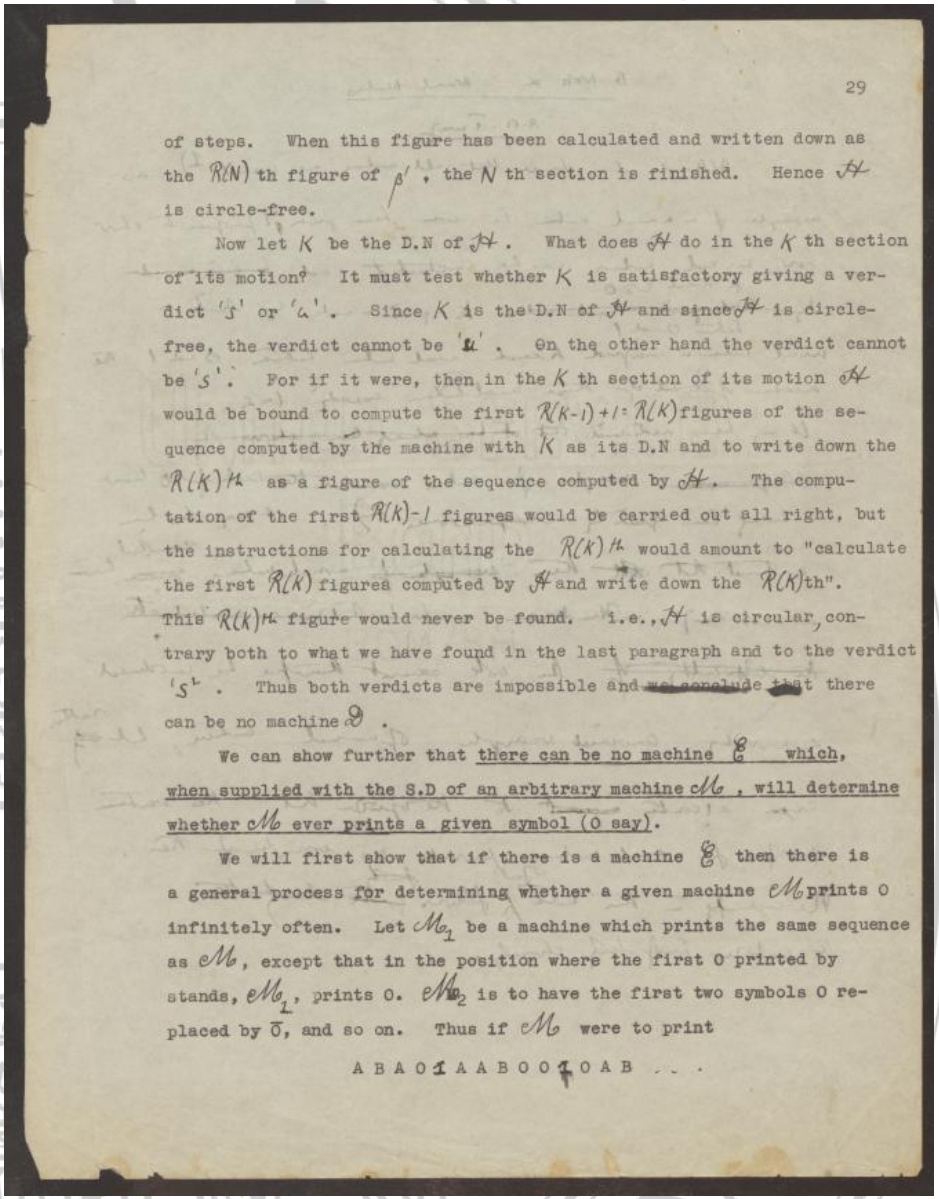
© P.N. Furbank, Turing Digital Archive

On Computable Numbers, with an Application to the Entscheidungsproblem, Proc. Lond. Math. Soc. (2) 42 pp. 230-265 (1936)

Turing intende dare una risposta negativa al problema della decisione

Ma centra il lavoro sulla nozione di calcolabile

Turing: Cosa significa calcolare?



29

of steps. When this figure has been calculated and written down as the $R(N)$ th figure of β', the N th section is finished. Hence $\mathcal{M}$ is circle-free.

Now let K be the D.N of $\mathcal{M}$. What does $\mathcal{M}$ do in the K th section of its motion? It must test whether K is satisfactory giving a verdict 's' or 'u'. Since K is the D.N of $\mathcal{M}$ and since $\mathcal{M}$ is circle-free, the verdict cannot be 'u'. On the other hand the verdict cannot be 's'. For if it were, then in the K th section of its motion $\mathcal{M}$ would be bound to compute the first $R(K-1)+1 = R(K)$ figures of the sequence computed by the machine with K as its D.N and to write down the $R(K)$th as a figure of the sequence computed by $\mathcal{M}$. The computation of the first $R(K)-1$ figures would be carried out all right, but the instructions for calculating the $R(K)$th would amount to "calculate the first $R(K)$ figures computed by $\mathcal{M}$ and write down the $R(K)$th". This $R(K)$th figure would never be found. i.e., $\mathcal{M}$ is circular, contrary both to what we have found in the last paragraph and to the verdict 's'. Thus both verdicts are impossible and we conclude that there can be no machine $\mathcal{D}$.

We can show further that there can be no machine $\mathcal{E}$ which, when supplied with the S.D of an arbitrary machine $\mathcal{M}$, will determine whether $\mathcal{M}$ ever prints a given symbol (0 say).

We will first show that if there is a machine $\mathcal{E}$ then there is a general process for determining whether a given machine $\mathcal{M}$ prints 0 infinitely often. Let $\mathcal{M}_1$ be a machine which prints the same sequence as $\mathcal{M}$, except that in the position where the first 0 printed by stands, $\mathcal{M}_1$, prints 0. $\mathcal{M}_2$ is to have the first two symbols 0 replaced by $\overline{0}$, and so on. Thus if $\mathcal{M}$ were to print

A B A 0 1 A A B 0 0 1 0 A B . . .

© P.N. Furbank, Turing Digital Archive

On Computable Numbers, with an Application to the Entscheidungsproblem, Proc. Lond. Math. Soc. (2) 42 pp. 230-265 (1936)

Turing intende dare una risposta negativa al problema della decisione

Ma centra il lavoro sulla nozione di calcolabile

Entscheidungsproblem
Problema della decisione

Esiste un metodo meccanico (cioè calcolabile)
per decidere se una qualunque formula logica è un teorema
(del calcolo del prim'ordine)?

[Hilbert & Ackermann, 1928]

Turing vuole dare una risposta *negativa* al problema della decisione

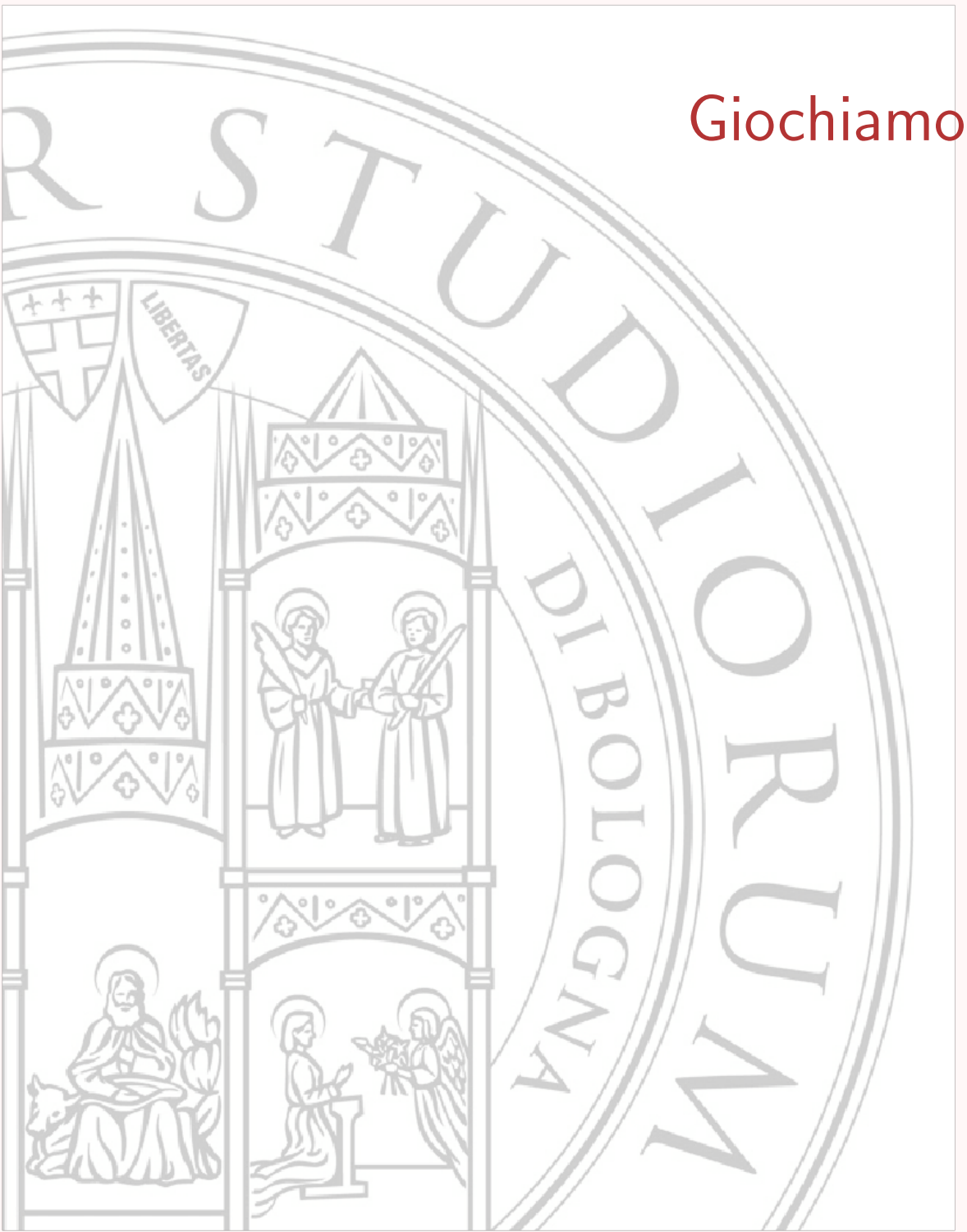
Ma centra il lavoro sulla nozione di *calcolabile*

Numeri (reali) calcolabili

- Parte decimale zero: interi.
- Parte decimale finita, o periodica: frazioni.
- Parte decimale infinita, non periodica ma ottenibile con un procedimento meccanico deterministico:

  tutti gli algebrici (p.e. $\sqrt{2}$, $\frac{1+\sqrt{5}}{2}$);
  certo *alcuni trascendenti* (p.e. π, e).

- Esistono numeri che *non* sono calcolabili?
  Cioè: la cui parte decimale non è ottenibile con un procedimento meccanico deterministico?



Giochiamo al. . . *computer!*

Giochiamo al. . . *computer!*

				2	7	3	+
				2	1	8	+
					1	1	=

Giochiamo al. . . *computer!*

				2	7	3	+
				2	1	8	+
					1	1	=

Giochiamo al. . . *computer!*

				2	7	**3**	+
				2	1	8	+
					1	1	=

Giochiamo al. . . *computer!*

				2	7	3	+
				2	1	**8**	+
					1	1	=

Giochiamo al. . . *computer!*

				2	7	3	+
				2	1	8	+
					1	**1**	=

Giochiamo al. . . *computer!*

				2	7	3	+
				2	1	8	+
					1	1	=
						2	

Giochiamo al. . . *computer!*

					1		
				2	7	3	+
				2	1	8	+
					1	1	=
						2	

Giochiamo al. . . *computer!*

					1		
				2	**7**	3	+
				2	1	8	+
					1	1	=
						2	

Giochiamo al. . . *computer!*

					1		
				2	7	3	+
				2	**1**	8	+
					1	1	=
						2	

Giochiamo al. . . *computer!*

					1		
				2	7	3	+
				2	1	8	+
					1	1	=
						2	

Giochiamo al. . . *computer!*

					1		
				2	7	3	+
				2	1	8	+
					1	1	=
					0	2	

Giochiamo al. . . *computer!*

				1	1		
				2	7	3	+
				2	1	8	+
					1	1	=
					0	2	

Giochiamo al. . . *computer!*

				1	1		
				2	7	3	+
				2	1	8	+
					1	1	=
					0	2	

Giochiamo al. . . *computer!*

				1	1		
				2	7	3	+
				2	1	8	+
					1	1	=
					0	2	

Giochiamo al. . . *computer!*

				1	1		
				2	7	3	+
				2	1	8	+
				0	1	1	=
					0	2	

Giochiamo al. . . *computer!*

				1	1		
				2	7	3	+
				2	1	8	+
				0	1	1	=
				5	0	2	

On computable numbers.

Leggiamo assieme. . .

- Computing is [. . .] done by writing [. . .] symbols on paper
- We may suppose this paper is divided into squares
- The behaviour of the computer [. . .] is determined by the symbols which he is observing, and his "state of mind"
- operations [are] split up into "simple operations"
- in a simple operation not more than one symbol is altered

On computable numbers, 2

Leggiamo assieme. . .

The most general single operation must therefore be taken to be one of the following:

1. A possible change of symbol together with a possible change of state of mind;
2. A possible change of observed squares, together with a possible change of state of mind.

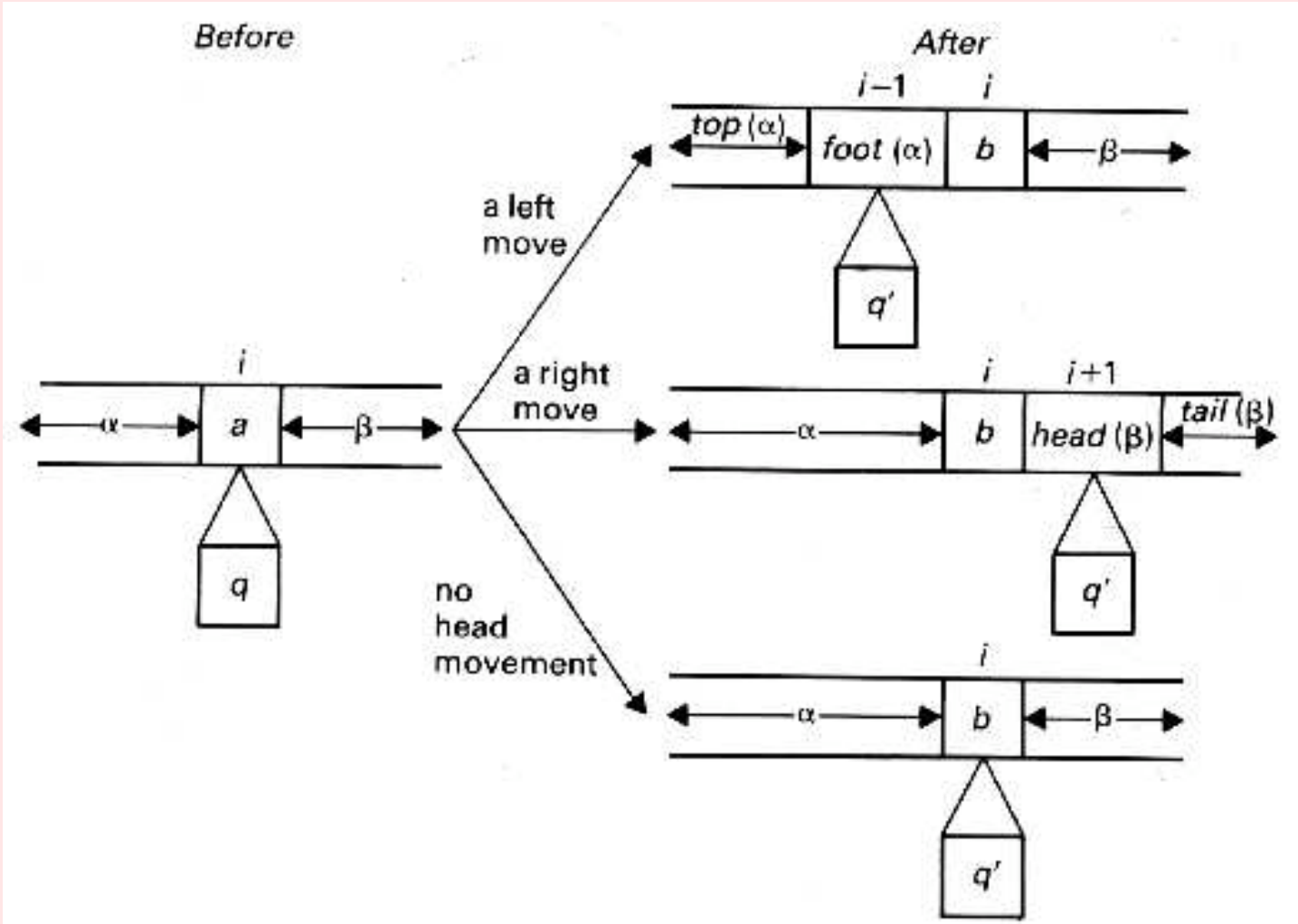
The operation actually performed is determined [. . .] by the state of mind of the computer and the observed symbols. In particular, they determine the state of mind of the computer after the operation is carried out.

Un commento un po' acido

Turing's "machines": These machines are humans who calculate.

[L. Wittgenstein,
Remarks on the Philosophy of Psychology, Vol. 1,
Blackwell, Oxford, 1980.]

La macchina di Turing



Le *quintuple*: (q, a, q', b, D), con $D \in \{\leftarrow, \rightarrow, -\}$

La macchina di Turing: esempio

				1			
				2	7	3	+
				2	1	8	+
					1	1	=
					0	2	

Una *quintupla*: (q, a, q', b, D):

1. q: ricordo parziale 3
2. a: vedo 2
3. q': ricordo parziale 5
4. b: lascio invariato 2
5. $D = \downarrow$: mi sposto una casella in basso

La macchina di Turing: esempio

				1			
				2	7	3	+
				2	1	8	+
				0	1	1	=
					0	2	

Una *quintupla*: (q, a, q', b, D):

1. q: ricordo parziale 5
2. a: vedo casella vuota
3. q': ricordo parziale 5
4. b: scrivo 0
5. $D = \downarrow$: mi sposto una casella in basso

La macchina di Turing, 2

- Alfabeto finito
- Stati interni in numero finito
- Operazioni: elementare symbol pushing
- Decisioni basate su informazioni locali, dunque finite
- "Nastro" potenzialmente infinito

- In una singola computazione: usata una porzione finita del nastro

- Descrizione finita di

  di una macchina (e del suo "programma")
  delle configurazioni di una singola computazione
- Dunque maneggiabile essa stessa da una macchina.

Una macchina che calcola

- Stato interno
- Posizione della testina
- Configurazione del nastro

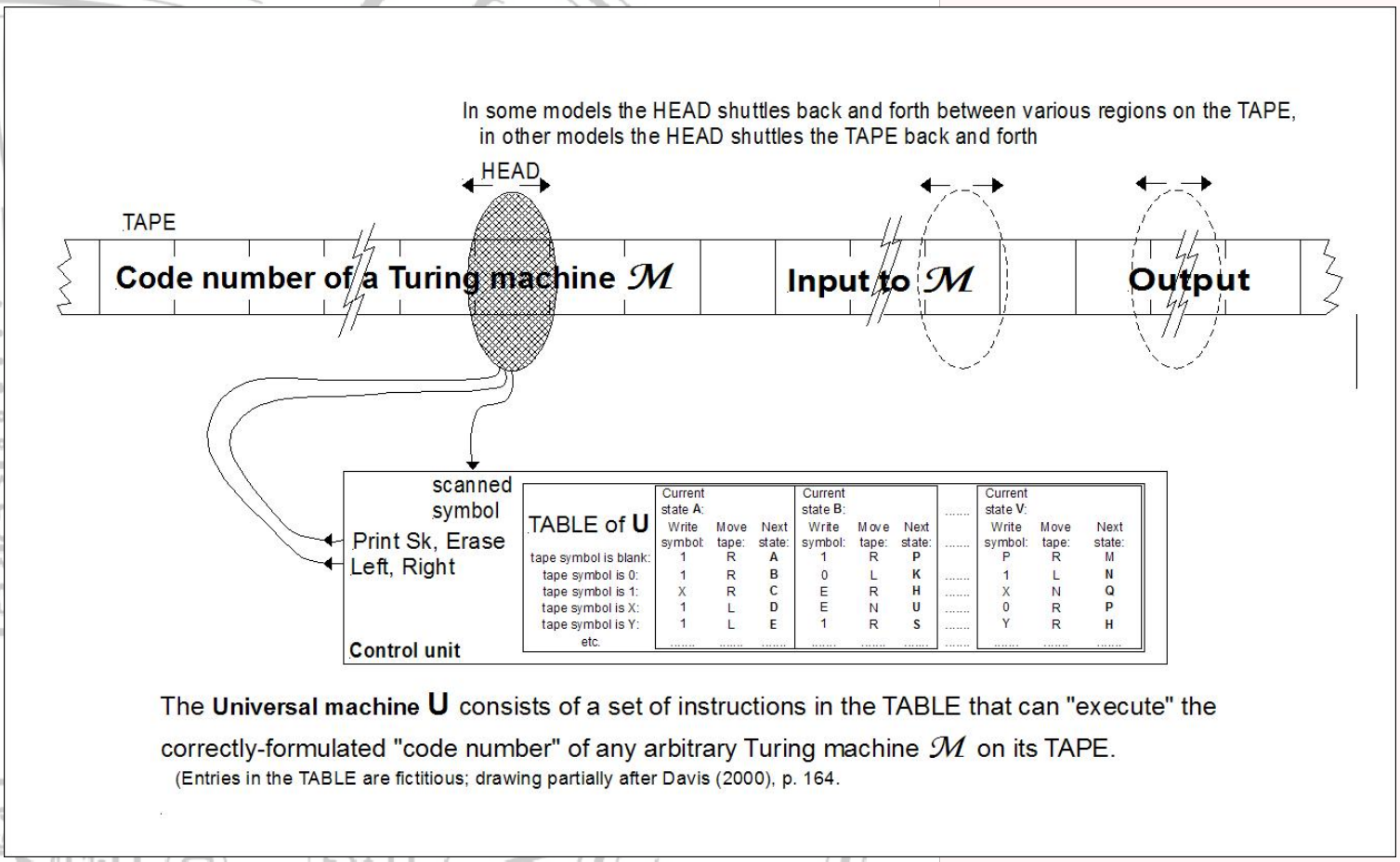
- Computazione: successione di configurazioni
- Se non vi sono quintuple applicabili, la macchina si ferma
- Computazioni finite e infinite

Configurazioni

The operation [...] is determined [...] by the state of mind of the computer and the observed symbols. In particular, they determine the state of mind of the computer after the operation is carried out.

- Una configurazione:
  - simboli sul nastro (in numero finito!)
  - posizione della testina
  - stato interno
- Una mossa: transizione da una configurazione ad un'altra
- determinata dalla configurazione precedente e le quintuple
- Il calcolo è completamente meccanico:
  è eseguibile da un *computer*!

La MdT universale



The **Universal machine U** consists of a set of instructions in the TABLE that can "execute" the correctly-formulated "code number" of any arbitrary Turing machine $\mathcal{M}$ on its TAPE.

(Entries in the TABLE are fictitious; drawing partially after Davis (2000), p. 164.

La macchina universale

- Un'unica macchina, U, simula tutte le altre
- Il suo "programma cablato" è un programma di emulazione della macchina che è il suo dato
- U è un calcolatore a programma memorizzato
- L'intuizione di von Neumann

Indice

Ricordate?
Calcolo e matematica: limitazioni

- Alcune costruzioni sono possibili
- Altre sono dimostrabilmente impossibili
- Con riga e compasso:

  ...

Negatives are such difficult things to prove.

[A. Christie, Three blind mice.]

Limitazione della Macchina di Turing?

- Esistono costruzioni non calcolabili da nessuna macchina?

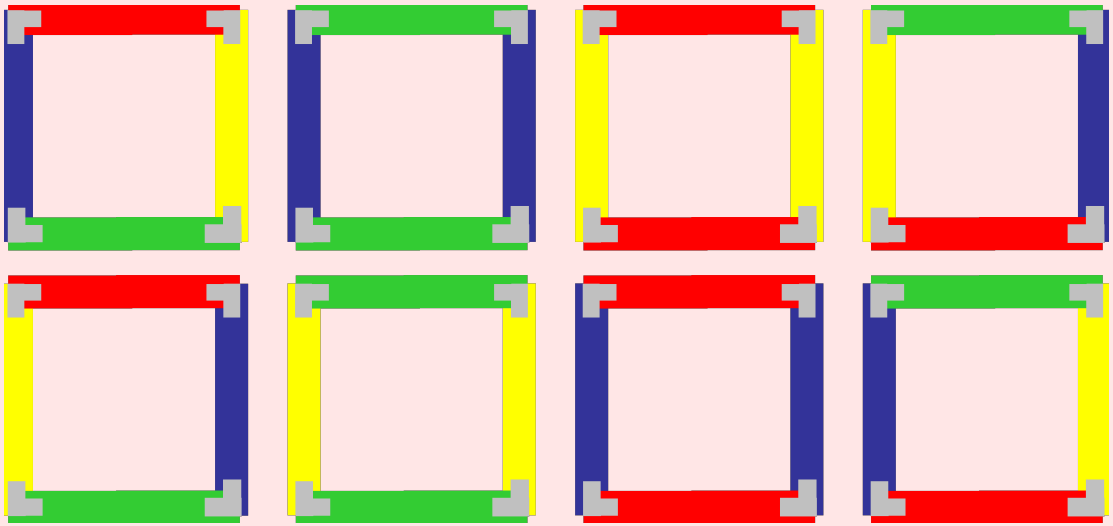
- Esistono numeri reali la cui espansione decimale non è prodotta da nessuna macchina
- Esistono problemi la cui soluzione non può essere calcolata: Problemi indecidibili.

Limitazione della Macchina di Turing?

- Esistono costruzioni non calcolabili da nessuna macchina?
- Esistono numeri reali la cui espansione decimale non è prodotta da nessuna macchina
- Esistono problemi la cui soluzione non può essere calcolata: Problemi indecidibili.

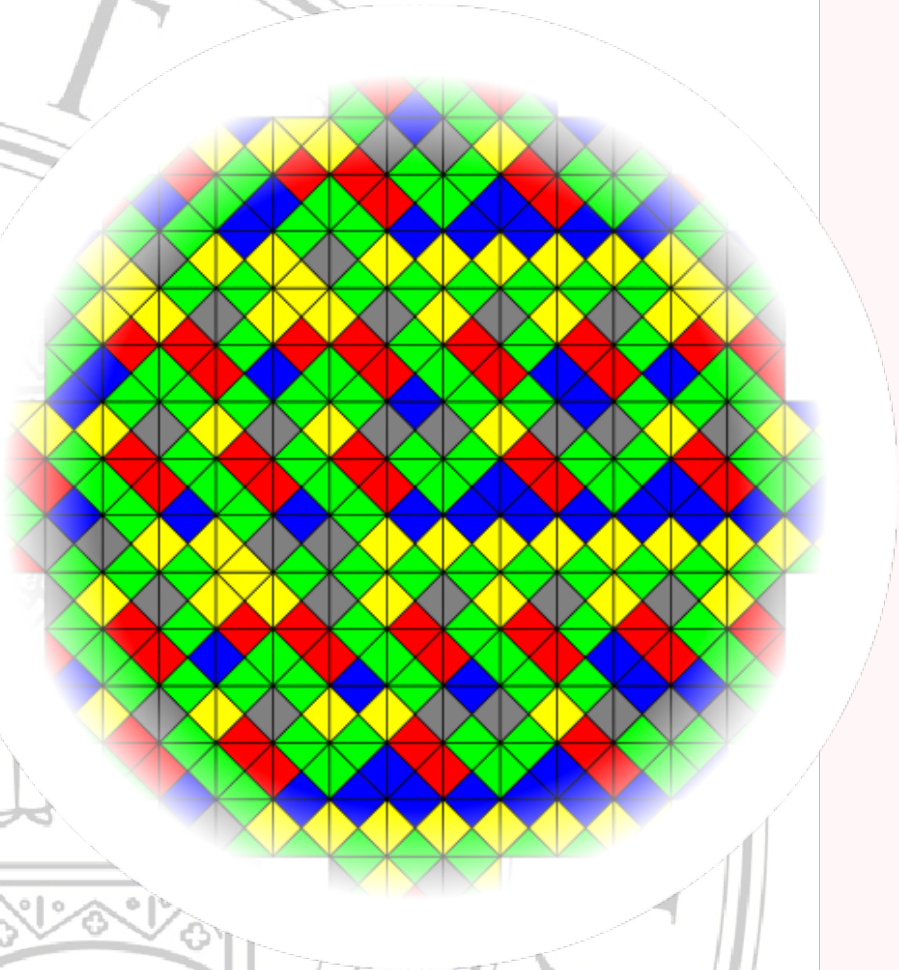
Un problema non decidibile

Piastrelle di Wang!



Dato un insieme di piastrelle,
decidere se possono piastrellare il piano

Ma allora?



Ricerca di specifiche soluzioni.

Non ci sono metodi generali

Altri problemi indecidibili

- Piastrellatura di Wang
- Problema della fermata
- Equivalenza dei programmi
- *Decimo problema di Hilbert*:

  Decidere se un'equazione a coefficienti interi ha una soluzione intera
- *Entscheidungsproblem*:

  Decidere se una qualunque formula logica è un teorema

Altri formalismi

- Le MdT sembrano rudimentali...
- Altri tentativi di formalizzare il concetto di "calcolabile"

  Turing: $\mathcal{T}$
  Kleene: $\mathcal{R}$
  Gödel: equazioni ricorsive
  Church: λ-calcolo
  Post: sistemi di riscrittura
  Kolmogorov
  ...
  e tutti i linguaggi di programmazione!

- Sono tutti equivalenti
- Giustificano il nome di *calcolabile* tout court.

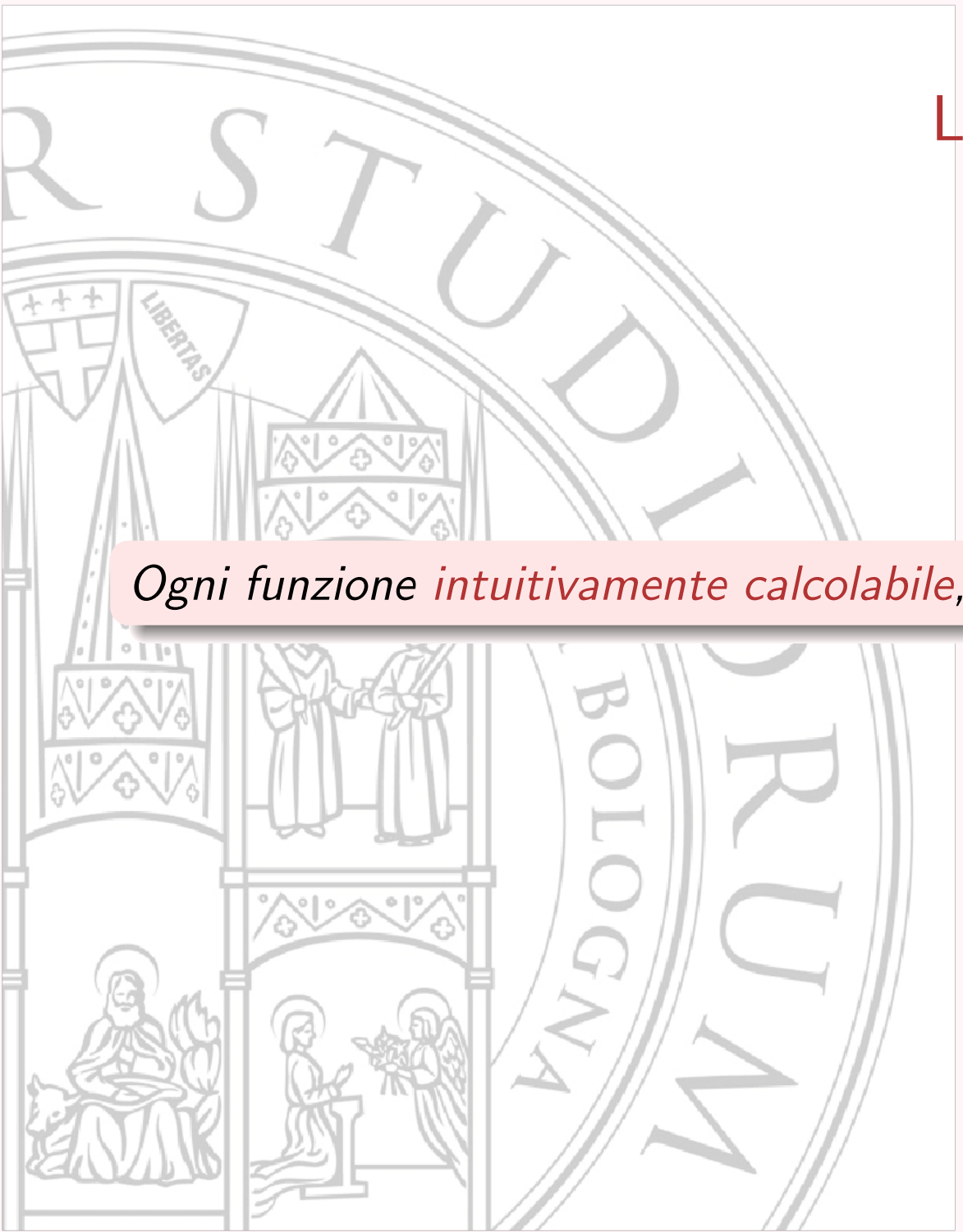
Equivalenze

Altri formalismi

- Le MdT sembrano rudimentali...
- Altri tentativi di formalizzare il concetto di "calcolabile"

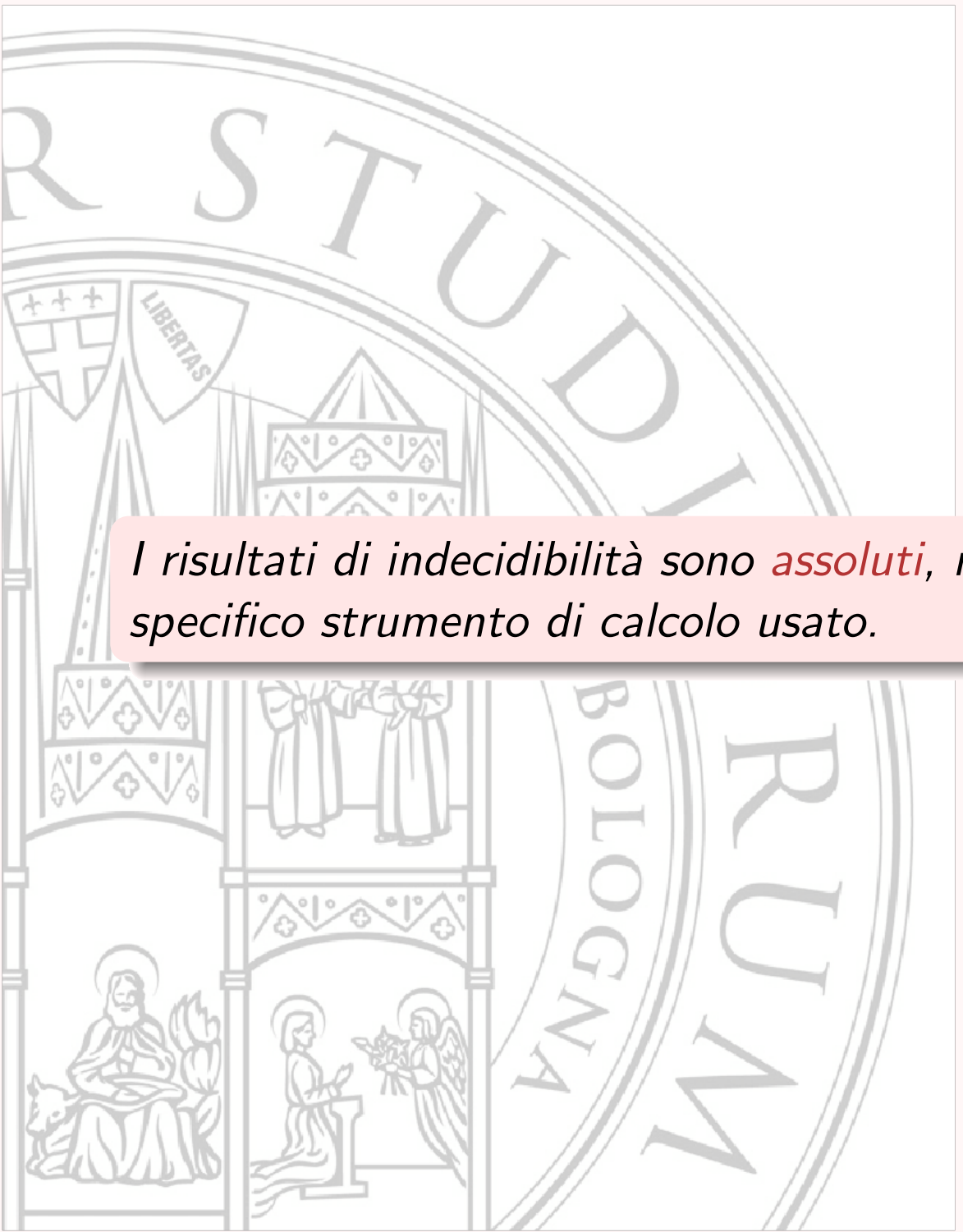
  Turing: $\mathcal{T}$
  Kleene: $\mathcal{R}$
  Gödel: equazioni ricorsive
  Church: λ-calcolo
  Post: sistemi di riscrittura
  Kolmogorov
  ...
  e tutti i linguaggi di programmazione!

- Sono tutti equivalenti
- Giustificano il nome di *calcolabile* tout court.



La tesi di Church

Ogni funzione intuitivamente calcolabile, è calcolabile da una MdT



Conseguenze

I risultati di indecidibilità sono assoluti, non dipendono dallo specifico strumento di calcolo usato.

Indice

Limiti nelle risorse

Un problema può essere risolubile...

ma richiedere:

- più tempo della vita dell'universo
- più spazio di lavoro del diametro stimato dell'universo
- più messaggi scambiati degli atomi dell'universo

Complessità computazionale

Classificare i problemi

in base

- alle risorse che sono necessarie;
- intrinsicamente;
- in funzione della dimensione dei dati

Esempi

- SORT: Ordinare n numeri, sfruttando confronti tra loro
  richiede $n \log n$ confronti
- SHORTEST PATH: Determinare il percorso minimo tra due città, assumendo distanze non negative
  richiede almeno un tempo polinomiale nel numero delle città (eg, n^2, algoritmo di Dijkstra)
- PRIMES: Determinare se un numero n è primo
  richiede tempo polinomiale nel numero di cifre di n
- WEAK MONADIC SECOND ORDER THEORY OF SUCCESSOR:
  richiede tempo esponenziale nella lunghezza della formula
- TRAVELLING SALESMAN: Date n città determinare un percorso (ciclico) che le tocca tutte una ed una sola volta
  richiede ??

Esempi

- SORT: Ordinare n numeri, sfruttando confronti tra loro
  richiede $n \log n$ confronti
- SHORTEST PATH: Determinare il percorso minimo tra due città, assumendo distanze non negative
  richiede almeno un tempo polinomiale nel numero delle città (eg, n^2, algoritmo di Dijkstra)
- PRIMES: Determinare se un numero n è primo
  richiede tempo polinomiale nel numero di cifre di n
- WEAK MONADIC SECOND ORDER THEORY OF SUCCESSOR:
  richiede tempo esponenziale nella lunghezza della formula
- TRAVELLING SALESMAN: Date n città determinare un percorso (ciclico) che le tocca tutte una ed una sola volta
  richiede ??

Esempi

- SORT: Ordinare n numeri, sfruttando confronti tra loro
  richiede $n \log n$ confronti
- SHORTEST PATH: Determinare il percorso minimo tra due città, assumendo distanze non negative
  richiede almeno un tempo polinomiale nel numero delle città (eg, n^2, algoritmo di Dijkstra)
- PRIMES: Determinare se un numero n è primo
  richiede tempo polinomiale nel numero di cifre di n
- WEAK MONADIC SECOND ORDER THEORY OF SUCCESSOR:
  richiede tempo esponenziale nella lunghezza della formula
- TRAVELLING SALESMAN: Date n città determinare un percorso (ciclico) che le tocca tutte una ed una sola volta
  richiede ??

Esempi

- SORT: Ordinare n numeri, sfruttando confronti tra loro
  richiede $n \log n$ confronti
- SHORTEST PATH: Determinare il percorso minimo tra due città, assumendo distanze non negative
  richiede almeno un tempo polinomiale nel numero delle città (eg, n^2, algoritmo di Dijkstra)
- PRIMES: Determinare se un numero n è primo
  richiede tempo polinomiale nel numero di cifre di n
- WEAK MONADIC SECOND ORDER THEORY OF SUCCESSOR:
  richiede tempo esponenziale nella lunghezza della formula
- TRAVELLING SALESMAN: Date n città determinare un percorso (ciclico) che le tocca tutte una ed una sola volta
  richiede ??

Esempi

- SORT: Ordinare n numeri, sfruttando confronti tra loro
  richiede $n \log n$ confronti
- SHORTEST PATH: Determinare il percorso minimo tra due città, assumendo distanze non negative
  richiede almeno un tempo polinomiale nel numero delle città (eg, n^2, algoritmo di Dijkstra)
- PRIMES: Determinare se un numero n è primo
  richiede tempo polinomiale nel numero di cifre di n
- WEAK MONADIC SECOND ORDER THEORY OF SUCCESSOR:
  richiede tempo esponenziale nella lunghezza della formula
- TRAVELLING SALESMAN: Date n città determinare un percorso (ciclico) che le tocca tutte una ed una sola volta
  richiede ??

Esempi

- SORT: Ordinare n numeri, sfruttando confronti tra loro
  richiede $n \log n$ confronti
- SHORTEST PATH: Determinare il percorso minimo tra due città, assumendo distanze non negative
  richiede almeno un tempo polinomiale nel numero delle città (eg, n^2, algoritmo di Dijkstra)
- PRIMES: Determinare se un numero n è primo
  richiede tempo polinomiale nel numero di cifre di n
- WEAK MONADIC SECOND ORDER THEORY OF SUCCESSOR:
  richiede tempo esponenziale nella lunghezza della formula
- TRAVELLING SALESMAN: Date n città determinare un percorso (ciclico) che le tocca tutte una ed una sola volta

  richiede ??

Esempi

- SORT: Ordinare n numeri, sfruttando confronti tra loro
  richiede $n \log n$ confronti
- SHORTEST PATH: Determinare il percorso minimo tra due città, assumendo distanze non negative
  richiede almeno un tempo polinomiale nel numero delle città (eg, n^2, algoritmo di Dijkstra)
- PRIMES: Determinare se un numero n è primo
  richiede tempo polinomiale nel numero di cifre di n
- WEAK MONADIC SECOND ORDER THEORY OF SUCCESSOR:
  richiede tempo esponenziale nella lunghezza della formula
- TRAVELLING SALESMAN: Date n città determinare un percorso (ciclico) che le tocca tutte una ed una sola volta

  richiede ??

Esempi

- SORT: Ordinare n numeri, sfruttando confronti tra loro
  richiede $n \log n$ confronti
- SHORTEST PATH: Determinare il percorso minimo tra due città, assumendo distanze non negative
  richiede almeno un tempo polinomiale nel numero delle città (eg, n^2, algoritmo di Dijkstra)
- PRIMES: Determinare se un numero n è primo
  richiede tempo polinomiale nel numero di cifre di n
- WEAK MONADIC SECOND ORDER THEORY OF SUCCESSOR:
  richiede tempo esponenziale nella lunghezza della formula
- TRAVELLING SALESMAN: Date n città determinare un percorso (ciclico) che le tocca tutte una ed una sola volta

  richiede ??

Esempi

- SORT: Ordinare n numeri, sfruttando confronti tra loro
  richiede $n \log n$ confronti
- SHORTEST PATH: Determinare il percorso minimo tra due città, assumendo distanze non negative
  richiede almeno un tempo polinomiale nel numero delle città (eg, n^2, algoritmo di Dijkstra)
- PRIMES: Determinare se un numero n è primo
  richiede tempo polinomiale nel numero di cifre di n
- WEAK MONADIC SECOND ORDER THEORY OF SUCCESSOR:
  richiede tempo esponenziale nella lunghezza della formula
- TRAVELLING SALESMAN: Date n città determinare un percorso (ciclico) che le tocca tutte una ed una sola volta

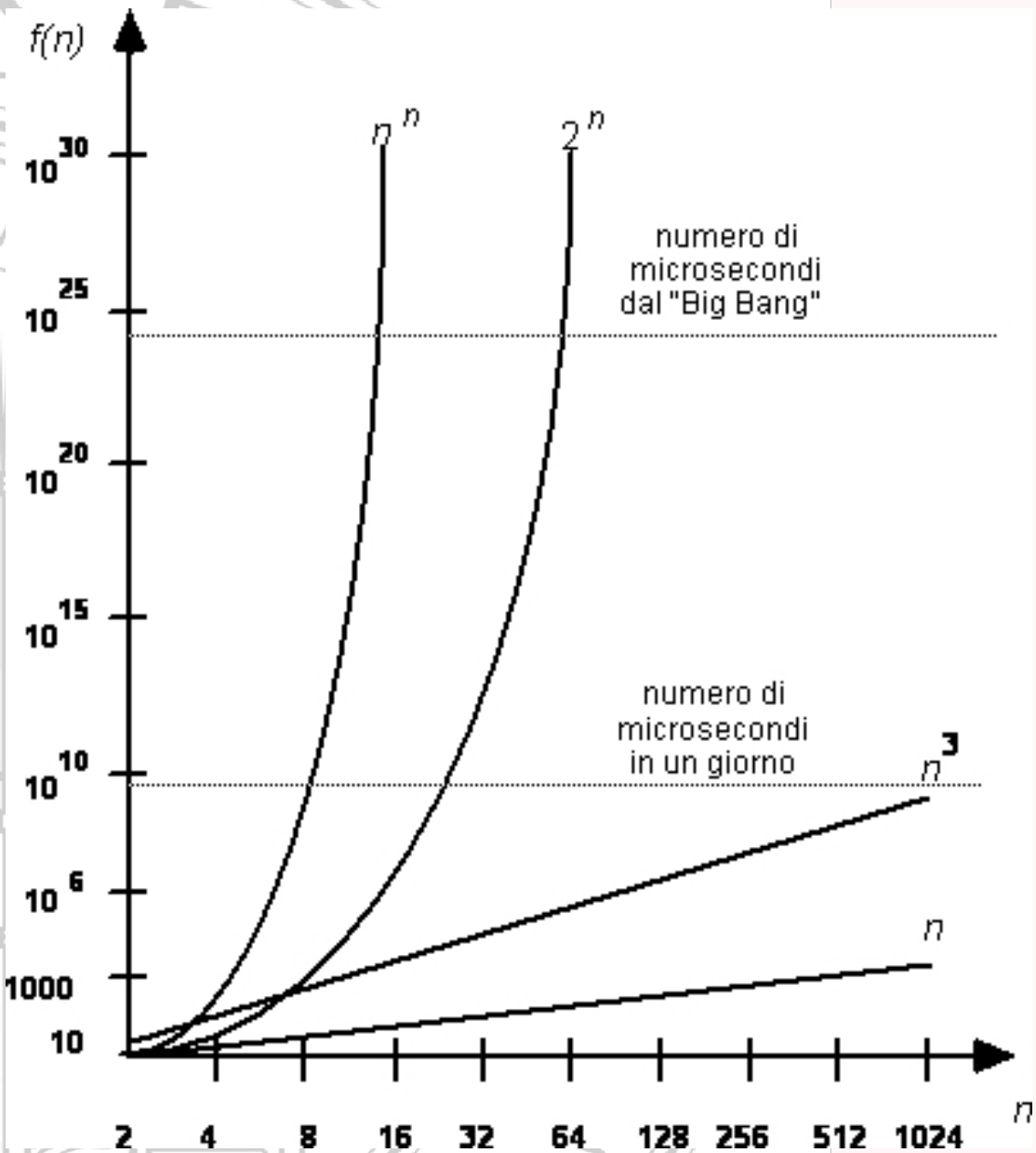
  richiede ??

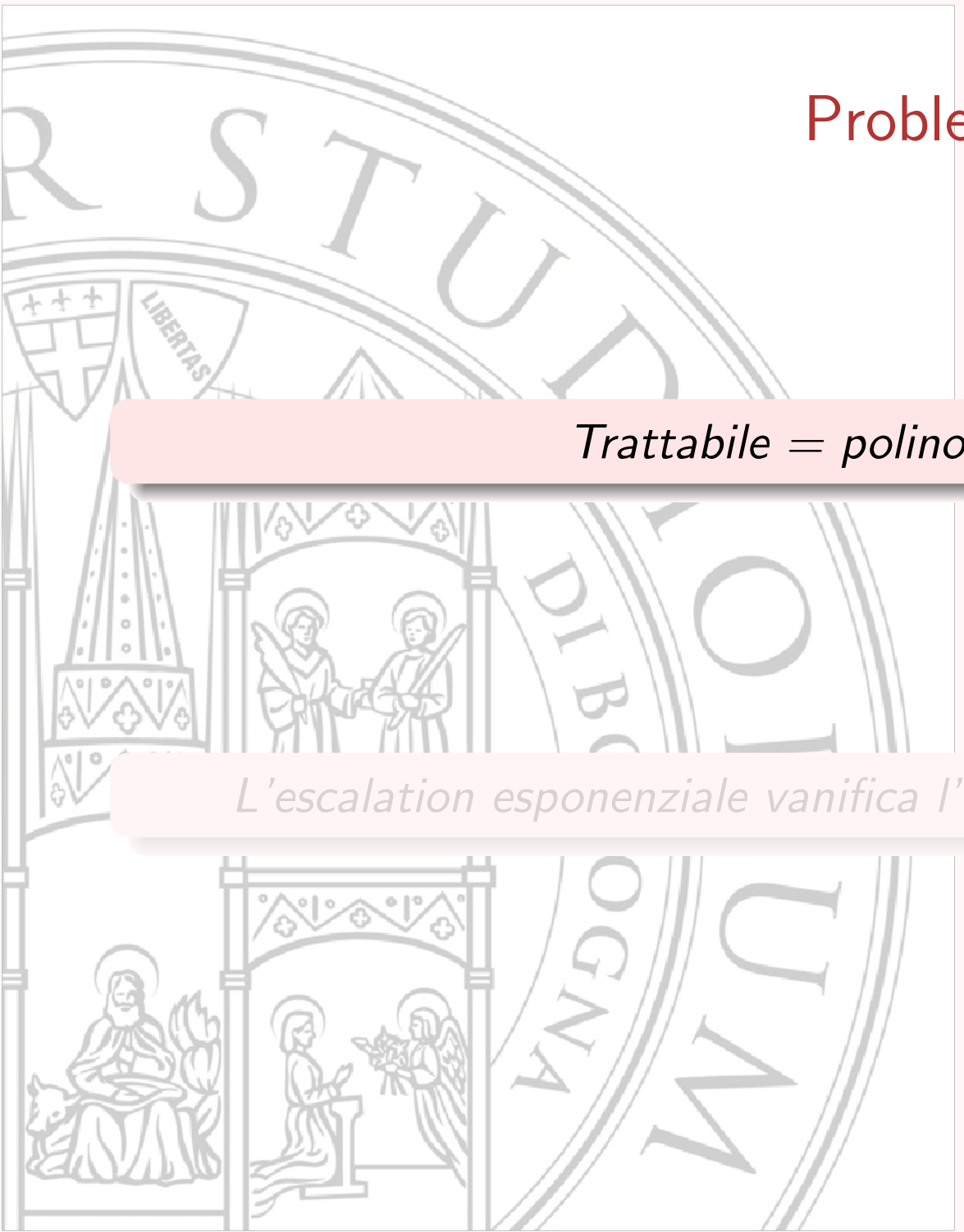
Esempi

- SORT: Ordinare n numeri, sfruttando confronti tra loro
  richiede $n \log n$ confronti
- SHORTEST PATH: Determinare il percorso minimo tra due città, assumendo distanze non negative
  richiede almeno un tempo polinomiale nel numero delle città (eg, n^2, algoritmo di Dijkstra)
- PRIMES: Determinare se un numero n è primo
  richiede tempo polinomiale nel numero di cifre di n
- WEAK MONADIC SECOND ORDER THEORY OF SUCCESSOR:
  richiede tempo esponenziale nella lunghezza della formula
- TRAVELLING SALESMAN: Date n città determinare un percorso (ciclico) che le tocca tutte una ed una sola volta

  richiede ??

Crescita delle funzioni





Problemi (in)trattabili

Trattabile = polinomiale

L'escalation esponenziale vanifica l'escalation tecnologica

Problemi (in)trattabili

Trattabile = polinomiale

L'escalation esponenziale vanifica l'escalation tecnologica

Problemi NP-completi

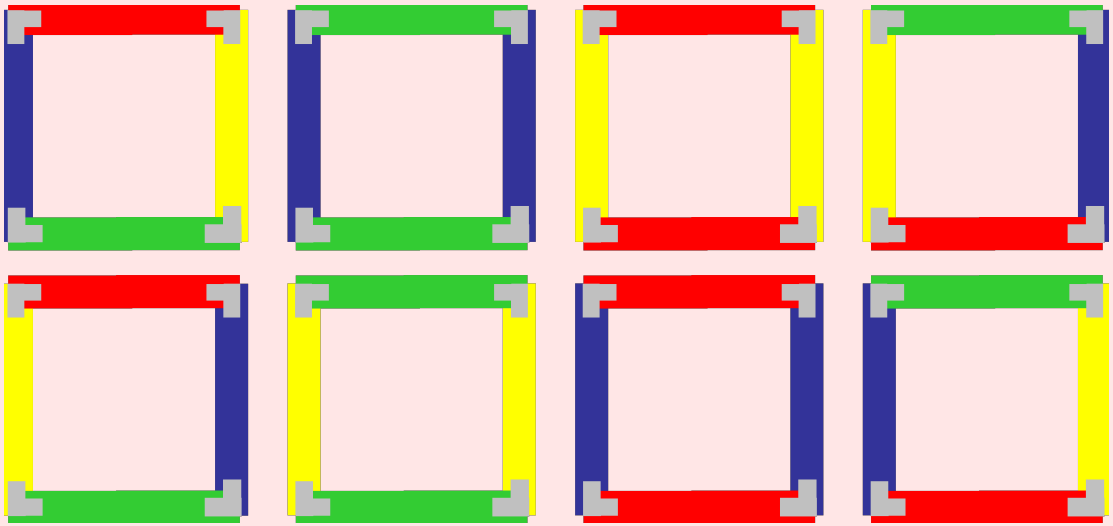
- Di alcuni problemi non conosciamo una limitazione intrinseca
- L'unico metodo che conosciamo è di forza bruta: esponenziale
- Ma non abbiamo dimostrazioni che non esistono algoritmi polinomiali
- Sappiamo riconoscere in modo efficiente se una soluzione candidata è davvero una soluzione
- TRAVELLING SALESMAN: Date n città determinare un percorso (ciclico) che le tocca tutte una ed una sola volta

Problemi NP-completi

- Di alcuni problemi non conosciamo una limitazione intrinseca
- L'unico metodo che conosciamo è di forza bruta: esponenziale
- Ma non abbiamo dimostrazioni che non esistono algoritmi polinomiali
- Sappiamo riconoscere in modo efficiente se una soluzione candidata è davvero una soluzione
- TRAVELLING SALESMAN: Date n città determinare un percorso (ciclico) che le tocca tutte una ed una sola volta

Un problema NP-completo

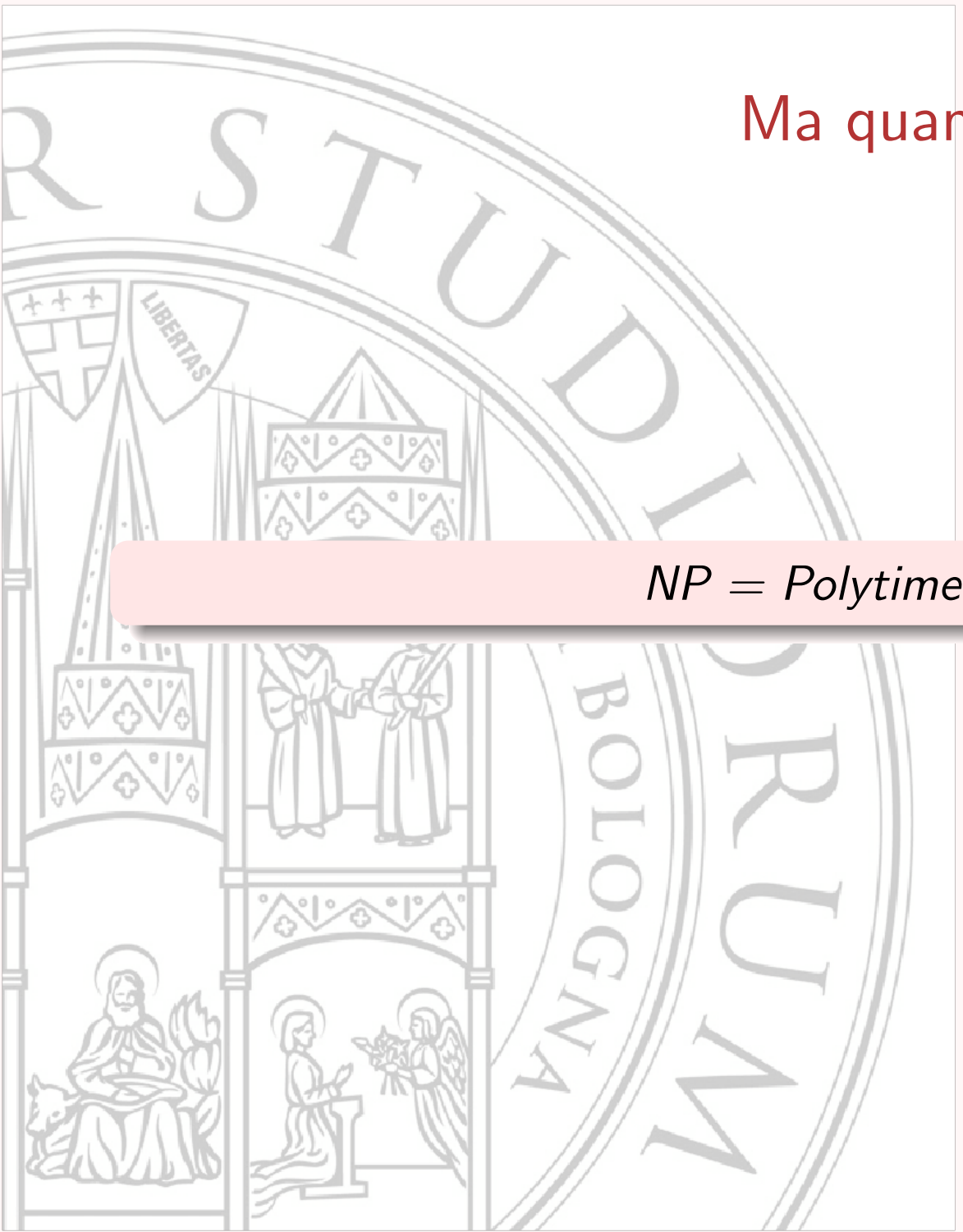
Piastrelle di Wang!



Dato un insieme di piastrelle,
decidere se possono piastrellare una porzione $n \times m$ di piano

Problemi di difficile classificazione

- Isomorfismo di grafi
- Fattorizzazione di un numero
- Non conosciamo algoritmi polinomiali, ma non sappiamo neppure classificarli come NP-completi



Ma quant'è difficile NP?

NP = Polytime ?

Indice

Che ci importa delle risorse?

- I computer sono sempre più veloci !
- La memoria non costa quasi nulla !

- Perché preoccuparci delle risorse?

Ci importa, eccome!

- Computer sempre più piccoli !
- Con batterie sempre più miniaturizzate !
- Sensori mobili (su segnali, boe, auto, lavatrici. . .)

- Soprattutto:
  L'esplosione esponenziale colpisce comunque !

Una prospettiva di ricerca

- Progettare linguaggi di programmazione
- Nei quali siano possibili solo programmi efficienti

- Non sono universali
- Ma *garantiscono* un consumo delle risorse appropriato

- Le logiche leggere

Indice

Tre *personæ*

- Un insieme di applicazioni
- Una tecnologia che rende possibili quelle applicazioni
- Una scienza che fonda quella tecnologia

Tre *personæ*

- Un insieme di applicazioni
- Una tecnologia che rende possibili quelle applicazioni
- Una scienza che fonda quella tecnologia

Tre *personæ*

- Un insieme di applicazioni
- Una tecnologia che rende possibili quelle applicazioni
- Una scienza che fonda quella tecnologia

Dovremmo...

Tre *personæ*

- Applicazioni
- Tecnologia
- Scienza

Saper usare le applicazioni

insieme

Al contesto e ai fondamenti scientifici nel quale esse si collocano

Per limitare l'obsolescenza
Per una piena cittadinanza

Concetti

Informatica:

- Studia i procedimenti effettivi di elaborazione dell'informazione.
- Contribuisce alle scienze con concetti propri, quali:
  - effettività
  - complessità computazionale
  - gerarchia di astrazione
  - informazione!
- Condivide con altre scienze:
  - interazione
  - comunicazione
  - problem solving

Ma soprattutto...

- Mette a disposizione strumenti linguistici
- Affinché ciò sia possibile e semplice
- Cioè evocativo, sintetico, economico

Ipsa forma est substantia

L'essenza dell'informatica risiede nell'immateriale dell'espressione linguistica del calcolo e dell'interazione.

Il modo di esprimere un concetto (un algoritmo, la struttura di un protocollo, un'architettura software) è altrettanto importante del concetto espresso.

Computational thinking

L'universo "è scritto in lingua matematica, e i caratteri son triangoli, cerchi ed altre figure geometriche"

Dopo Turing, quella lingua è anche una lingua computazionale, che permette di raggiungere livelli di espressione ogni giorno più sofisticati.