Come contano i roditori
(seconda parte)

di **Isabelita Antolini**
Stefano Leonesi
Carlo Toffalori

La prima parte di questo articolo è stata pubblicata sul n. 56 della "Lettera". Adesso vedremo se è vero che i castori sanno calcolare meglio degli esseri umani. Il dubbio viene da una funzione che forse solo i migliori roditori sanno computare e dalle difficoltà che nascono a cercare di imitarli.

Alberi di Natale e draghi senza coda

Nella prima parte di questo articolo abbiamo discusso l'esistenza di funzioni prive di un algoritmo di calcolo e abbiamo presentato due esempi al proposito: le funzioni Σ e S di Rado, che associano ad ogni naturale, rispettivamente, il massimo punteggio nel gioco del castoro laborioso e il massimo numero di mosse ammissibili. Abbiamo, anzi, dato una dimostrazione rigorosa dell'impossibilità di costruire procedimenti che computino Σ e S (ai sensi della *tesi di Turing*).
In questa seconda parte, discutiamo le difficoltà pratiche del calcolo di Σ e S. Assumiamo che il lettore abbia familiarità con la prima parte. Come già rilevato a proposito di $\Sigma(2)$, l'ostacolo principale alla computazione di Σ e S è che ci sono *troppi* concorrenti ammessi ai relativi giochi e dunque *troppe* macchine da controllare prima di stabilire i punteggi vincenti. Vediamo infatti di stimarne preventivamente il numero per un dato naturale n (generalizzando le osservazioni fatte per $n = 2$).

Gli autori

Isabelita Antolini, Stefano Leonesi e **Carlo Toffalori** fanno parte del gruppo di Logica matematica del Dipartimento di Matematica e Informatica dell'Università di Camerino. Leonesi e Toffalori sono autori di due libri, recentemente editi dalla Springer: *Un invito all'algebra* e *Numeri e crittografia*.

Contiamo in particolare le macchine di Turing su {1} a $n+1$ stati $q(0), q(1), \ldots, q(n)$ che sono prive di istruzioni relative a $q(n)$ e quindi sono potenzialmente ammesse ai giochi di $\Sigma(n)$ e $S(n)$ in quanto capaci di soddisfare almeno il requisito (ii). Alcune di queste macchine M possono eventualmente mancare di istruzioni anche su altre coppie $(0, q(i))$ o $(1, q(i))$ con $i < n$. D'altra parte, l'assenza di queste istruzioni non inficia la computazione della macchina sullo schermo bianco, e dunque la sua ammissione ai giochi, ed i punteggi da essa ottenuti, che dipendono solo dalle istruzioni effettivamente presenti. Possiamo dunque aggiungere a M nuove istruzioni su ogni possibile coppia prima dimenticata $(0, q(i))$, $(1, q(i))$ con $i < n$, magari comandandole di scrivere 1, spostarsi a destra ed entrare nello stato $q(n)$ (dopo di che, M è obbligata ad arrestarsi). Otteniamo una nuova macchina M', ancora ammessa ai giochi per n e anzi capace di ottenere, tanto per Σ quanto per S, punteggio uguale o al più superiore di 1 rispetto a M. Così, nella ricerca dei risultati vincenti, M può essere degnamente rappresentata da M'. In altre parole, non è restrittivo limitare la nostra analisi alle macchine di Turing negli stati $q(0), q(1), \ldots, q(n)$ che sono prive di istruzioni per $q(n)$, ma hanno disposizioni precise per ogni ulteriore coppia $(0, q(i))$ e $(1, q(i))$ per $i < n$.
Le macchine così ottenute corrispondono alle funzioni *totali*:
il cui dominio è l'insieme delle coppie che hanno 0 o 1 come prima componente e $q(0)$ o $q(1)$ o … o $q(n-1)$ (ma non $q(n)$) come seconda;
la cui immagine è composta da terne che hanno ancora 0 o 1 come prima componente, L o R come seconda, e am-

mettono stavolta *q(0), q(1), …* o anche *q(n)* come terza componente.

Dunque, si parte da un dominio composto da $2n$ coppie e si associa ad ognuna di esse una tra $4(n+1)$ possibili terne. Le funzioni *totali* che si ottengono in tal modo (e dunque le macchine potenzialmente ammesse ai giochi per n) sono $(4(n+1))^{2n}$.

Il valore che questa espressione assume per $n = 2, 3, 4$ è già tale da scoraggiare ogni analisi. Le macchine da controllare sono, infatti,

per $n = 2$: $12^4 = 20.736$,
per $n = 3$: $16^6 = 16.777.216$,
per $n = 4$: $20^8 = 25.600.000.000$.

Ci sono tuttavia alcune osservazioni che permettono di ridurre, almeno parzialmente, il gran numero di casi da esaminare.

1 Per esempio, chiamiamo due macchine M e M' *simmetriche* quando ammettono le stesse istruzioni, salvo che ogni comando L in M diventa R in M' e viceversa. Così, se M ha l'istruzione *(1, q(0)) → (0, L, q(0))*, allora M' contiene *(1, q(0)) → (0, R, q(0))*; se M ha *(1, q(1)) → (1, R, q(3))*, allora M include *(1, q(1)) → (1, L, q(3))* e così via. Le computazioni di M' sono le stesse di M, salvo che si sviluppano specularmente a destra (o sinistra) quando quelle corrispondenti di M si muovono verso sinistra (o destra, rispettivamente). È chiaro che, se M e M' giocano per $\Sigma(n)$ o per $S(n)$, allora ottengono lo stesso risultato. Dunque possiamo limitarci a considerare una delle due per il calcolo di $\Sigma(n)$ e $S(n)$. In particolare, possiamo ammettere di trattare macchine di Turing a $n+1$ stati che, in aggiunta alle condizioni (i) e (ii), soddisfano anche quella di includere un'istruzione della forma *(0, q(0)) → (…, R, …)*, cioè di muoversi verso destra nel primo passo della computazione sullo schermo bianco: ogni altra macchina dimenticata è simmetrica di una di queste.

2 Consideriamo adesso il caso di macchine M a $n+1$ stati che, per ogni $i < n$, contengono istruzioni della forma *(0, q(i)) → (0, …, …)*. Esaminiamo la computazione di M sullo schermo bianco. M non ha istruzioni che le permettano in queste condizioni di scrivere 1, dunque diverge (ed è dunque squalificata nei giochi di $\Sigma(n)$ e $S(n)$) oppure, se converge, ammette punteggio 0. Anzi, nella seconda eventualità, deve cambiare stato ad ogni passo della computazione (una ripetizione di stato la porta a divergere) e dunque termina il suo lavoro entro $n+1$ passi. Non vale quindi la pena di attardarsi a considerare queste situazioni e conviene conseguentemente limitare l'analisi a macchine M a $n+1$ stati che, per qualche $i < n$, contengono l'istruzione *(0, q(i)) → (1, …, …)*. Anzi, salvo permutare tra loro gli stati di M, possiamo addirittura ammettere $i = 0$, cioè che M abbia l'istruzione *(0, q(0)) → (1, …, …)*. Infatti, tutte quelle macchine cui ci siamo appena ristretti giungono a stampare un 1 dopo al più n passi. Trascurare questa parte iniziale della loro computazione ed ammettere di scrivere direttamente 1 sullo schermo non altera il punteggio della macchina nel gioco di $\Sigma(n)$ e abbassa al più di n il suo risultato nel gioco di $S(n)$. Quindi, possiamo considerare solo macchine contenenti l'istruzione *(0, q(0)) → (1, …, …)* e, anzi, ricordando la precedente osservazione, *(0, q(0)) → (1, R, …)*.

3 Possiamo anche escludere *(0, q(0)) → (1, R, q(0))* che porta M a divergere, e *(0, q(0)) → (1, R, q(n))*, che le assegna punteggio 1 in Σ e in S, e dunque ammettere che M includa *(0, q(0)) → (1, R, q(i))* per qualche i con $0 < i < n$. Anzi, procedendo come in *2*, possiamo supporre $i = 1$ e dunque che M contenga *(0, q(0)) → (1, R, q(1))*.

4 È forse utile ricordare che le macchine M che ci interessano devono comunque arrestarsi quando partono dell'*input* bianco. D'altra parte, si ha convergenza solo in assenza di istruzioni da eseguire ed abbiamo precedentemente ammesso istruzioni su ogni stato *q(i)* con $i < n$. Così l'unico stato che permette a M di arrestarsi è *q(n)* e quindi almeno un'istruzione di M deve richiedere di entrare nello stato *q(n)* (e magari di stampare contemporaneamente un ultimo 1 per accrescere il punteggio per $\Sigma(n)$, e di muoversi verso destra R, visto che l'ultimo spostamento prima della fermata è assolutamente ininfluente per l'esito della computazione). Naturalmente, può accadere che M contenga varie istruzioni di questo tipo, tali dunque da condurla a *q(n)* in più possibili situazioni. Ma solo una di queste istruzioni sarà usata nella computazione sull'*input* bianco, visto che dopo la sua esecuzione M si arresta; dunque possiamo ammettere che ci sia esattamente una istruzione del tipo sopra descritto in M.

5 Possiamo escludere che l'istruzione di cui al punto 4 riguardi *(0, q(1))*. Altrimenti la macchina M, subito dopo aver eseguito al primo passo della computazione sullo schermo bianco l'istruzione *(0, q(0)) → (1, R, q(1))*, si trova ad applicare *(0, q(1)) → (1, R, q(n))* e dunque a convergere in 2 passi stampando 2 volte 1, dunque con punteggio 2 tanto per $\Sigma(n)$ che per $S(n)$.

Per apprezzare le semplificazioni che le precedenti considerazioni assicurano, consideriamo il caso $\boldsymbol{n = 2}$, già parzialmente anticipato nella prima parte. Le macchine in esame M hanno soltanto lo stato *q(1)* in aggiunta a *q(0)* e *q(n) = q(2)*. Dunque, per le osservazioni 1, 2 e 3, contengono esplicitamente l'istruzione *(0, q(0)) → (1, R, q(1))*, senza bisogno di permutazione degli stati. Conseguentemente, al

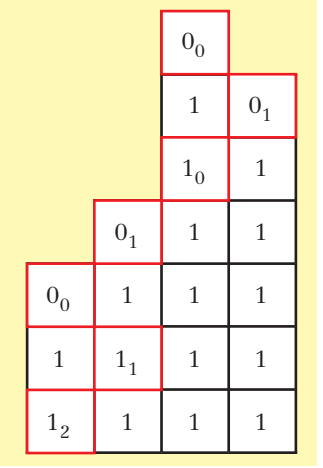
primo passo della computazione stampano 1, si muovono verso destra ed incontrano il simbolo 0 nello stato *q(1)*. Un'eventuale istruzione *(0, q(1)) → (…, …, q(2))* è esclusa da 5. L'osservazione 4 permette allora di distinguere due casi possibili (tra loro alternativi):
a) *M* contiene *(1, q(0)) → (1, R, q(2))*;
b) *M* contiene *(1, q(1)) → (1, R, q(2))*.
Le macchine corrispondenti ad *a)* si possono suddividono ulteriormente in base alle istruzioni riguardanti le coppie *(0, q(1))* e *(1, q(1))*; le terne corrispondenti possono ammettere 0 o 1 come prima coordinata, *L* o *R* come seconda, *q(0)* o *q(1)* (ma non *q(2)*) come terza. Si individuano così 8 possibili casi per ciascuna coppia e, in definitiva, 64 possibili macchine da esaminare. Altrettanto accade per *b)*. Si verifica dunque che l'analisi per il calcolo di $\Sigma(2)$ e $S(2)$ si restringe a 128 macchine, ciascuna ovviamente con 3 stati. La situazione che ne deriva è ragionevolmente gestibile, e permette di concludere:

$\Sigma(2) = 4$, $S(2) = 6$.

Omettiamo i dettagli dei calcoli che conducono a questa conclusione (il lettore che ha curiosità e pazienza sufficienti, e un pomeriggio di tempo libero da trascorrere, può anche svolgerli da solo). Proponiamo semmai un esempio di una macchina di Turing *M* a 3 stati che partecipa ai giochi di $\Sigma(2)$ e $S(2)$ ed ottiene i punteggi vincenti. *M* ha le seguenti istruzioni:

(0, q(0)) → (1, R, q(1)), (1, q(0)) → (1, L, q(1)),
(0, q(1)) → (1, L, q(0)), (1, q(1)) → (1, L, q(2)).

La sua computazione sull'*input* bianco, descritta nel modo spiegato nel paragrafo 1 (per la macchina che fa l'addizione), è la seguente:



Come si vede, *M* ottiene punteggio 4 in 6 spostamenti.

Quando però si considerano naturali $n > 2$, le cose si complicano radicalmente e troppo numerosi sembrano i casi da doversi contemplare. Al proposito, lo stesso Rado esprimeva un profondo pessimismo nel 1963, quando scriveva in [4]: "*anche se abili matematici ed esperti programmatori hanno provato a valutare Σ(3) e S(3), non c'è nessuna evidenza che gli approcci conosciuti possano fornire una risposta, neppure con l'aiuto di calcolatori ad alta velocità e programmi elaborati. Per quanto riguarda poi Σ(4) e S(4), allo stato attuale la situazione sembra essere assolutamente senza speranze.*"

Eppure, almeno per $n = 3$ e 4, l'opinione di Rado si è rivelata eccessivamente negativa. Già nel 1965, Rado medesimo, con il suo allievo Lin [3], riusciva a computare:

$\Sigma(3) = 6$, $S(3) = 21$

Mentre, per $n = 4$, si è dovuto attendere il 1983 e il lavoro di Allen Brady [1] per sapere:

$\Sigma(4) = 13$, $S(4) = 107$.

Ambedue i risultati richiedono, come previsto da Rado, sia abilità matematica che un sostanziale aiuto da parte dei calcolatori (in dosi ovviamente maggiori per 4). Accenniamo brevemente ai punti fondamentali delle relative analisi.
Cominciamo ovviamente dal caso più "facile", quello di:

n = 3.

Con l'uso delle osservazioni *1-5*, si individuano quattro possibili coppie che conducono alla terna di arresto *(1, R, q(3))*:

(1, q(0)), (1, q(1)), (1, q(2)), (0, q(2)).

Si distinguono, corrispondentemente, 4 classi di macchine di Turing:
quelle della prima classe contengono l'istruzione *(1, q(0)) → (1, R, q(3))*;
quelle della seconda contengono l'istruzione *(1, q(1)) → (1, R, q(3))*;
quelle della terza contengono l'istruzione *(1, q(2)) → (1, R, q(3))*;
quelle dell'ultima, infine, contengono *(0, q(2)) → (1, R, q(3))*.
Il guaio è che, stavolta, ogni classe include un numero di macchine ben maggiore di 64. In ciascun caso ci sono infatti 4 coppie in attesa di istruzioni (per esempio, *(0, q(1)),*

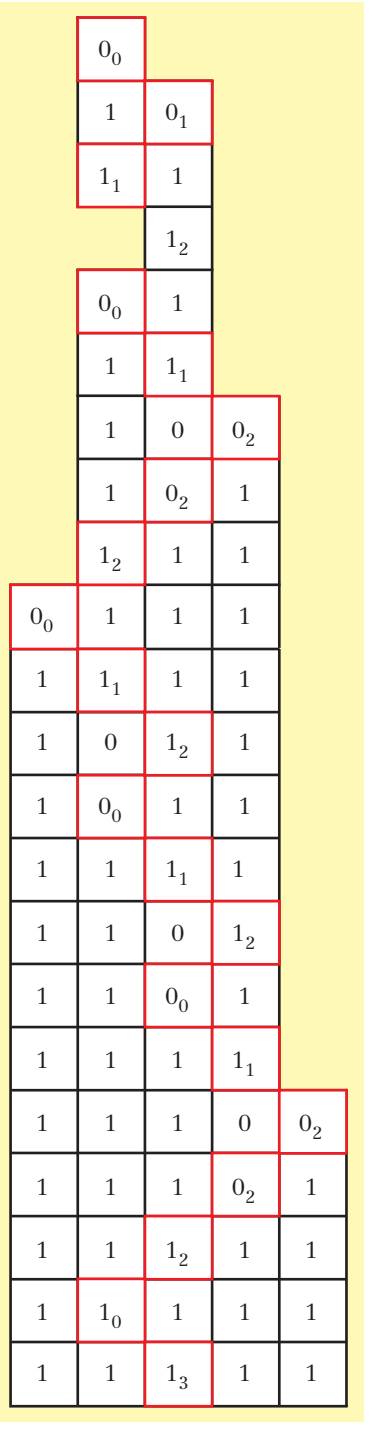
(0, q(2)), (1, q(1)), (1, q(2)) per la prima classe); le terne, cui queste coppie sono riconducibili, possono ammettere 0 o 1 come prima componente, *L* o *R* come seconda, *q(0)* o *q(1)* o *q(2)* (ma non *q(3)*) come terza. Ogni coppia ha, quindi, 2 × 2 × 3 = 12 immagini possibili; complessivamente, si determinano allora in ciascuna delle 4 classi 12^4=20.736 macchine e quindi, in totale, 4 × 20.736 = 82.944 macchine partecipanti ai giochi di Σ(3) e di *S*(3) anche dopo le semplificazioni sopra illustrate: un numero di campioni largamente inferiore ai 16.777.216 iniziali e tuttavia ancora cospicuo e difficile da dominare.

Comunque l'analisi di Rado e Lin si sviluppa proprio da questa base. Per ciascuna delle 4 classi sopra individuate, si distinguono le 12 possibili immagini di ciascuna delle 4 coppie diverse da *(0, q(0))* e da quella che conduce alla configurazione finale con lo stato *q(3)*. Un primo rozzo esame dei vari casi possibili identifica alcuni comportamenti chiave, che vengono verificati al calcolatore mediante opportuni programmi. Questa prima ispezione induce a congetturare *S*(3) = 21. Si costruiscono conseguentemente programmi per generare le varie macchine coinvolte e controllarne la computazione per i primi 21 passi. Di quelle che convergono entro tale termine, si annota il punteggio per Σ (e per *S*): in particolare, si trovano 5 macchine che ottengono risultato 6 nel gioco di Σ(3), impiegando un numero di spostamenti inferiore a 21 (soltanto 11 in un caso), e un'altra macchina che si ferma dopo 21 passi con punteggio 5. Per le altre macchine – quelle che non si arrestano entro 21 spostamenti – si cerca di identificare una qualche periodicità nella computazione, il ritorno cioè dopo qualche passo ad una stessa configurazione. Un tale comportamento ripetitivo implica ovviamente la divergenza della macchina (e dunque la sua squalifica nei due giochi).

Dopo questa selezione, restano soltanto 40 macchine *residue* da considerare, quelle per cui l'analisi dei primi 21 passi non getta sufficiente luce sul comportamento complessivo. Ma prima di avventurarci nell'ultima parte dell'indagine, riportiamo alcuni esempi relativi ai casi "positivi" appena trattati. Il lettore sarà infatti curioso di conoscere quale è la macchina a 4 stati che ha punteggio vincente per *S*(3), quella che converge solo dopo 21 spostamenti. Ebbene, la macchina in questione è quella *M* che ha istruzioni:

(0, q(0)) → *(1, R, q(1))*, *(1, q(0))* → *(1, R, q(3))*,
(0, q(1)) → *(1, L, q(1))*, *(1, q(1))* → *(0, R, q(2))*,
(0, q(2)) → *(1, L, q(2))*, *(1, q(2))* → *(1, L, q(0))*,

La sua computazione sullo schermo bianco, descritta nel modo spiegato nel paragrafo 1, è quella della figura a fianco:

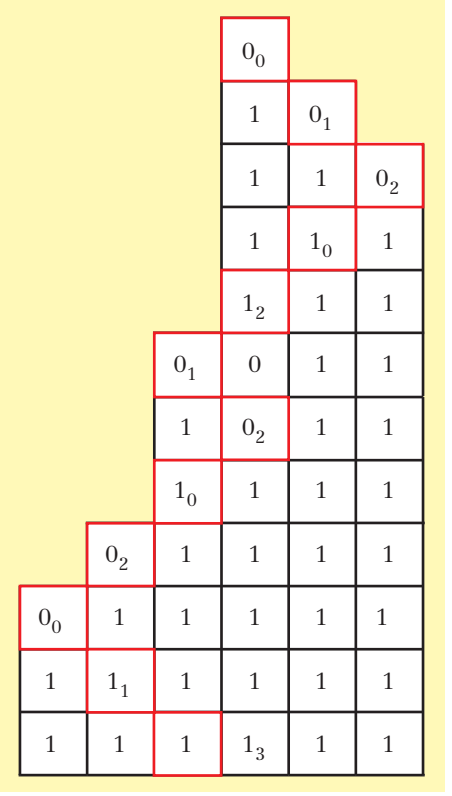


Come si vede, *M* ottiene solo punteggio 5 per il gioco di Σ(3) ma impiega un numero di spostamenti massimo 21 (si contino le righe della computazione).

Invece la macchina che ottiene punteggio vincente 6 per Σ(3), ma necessita di soltanto 11 spostamenti, è quella che ha istruzioni:

(0, q(0)) → (1, R, q(1)), (1, q(0)) → (1, L, q(2)),
(0, q(1)) → (1, R, q(2)), (1, q(1)) → (1, R, q(3)),
(0, q(2)) → (1, L, q(0)), (1, q(2)) → (0, L, q(1)).

La sua computazione sullo schermo bianco, descritta nel modo consueto, è la seguente:

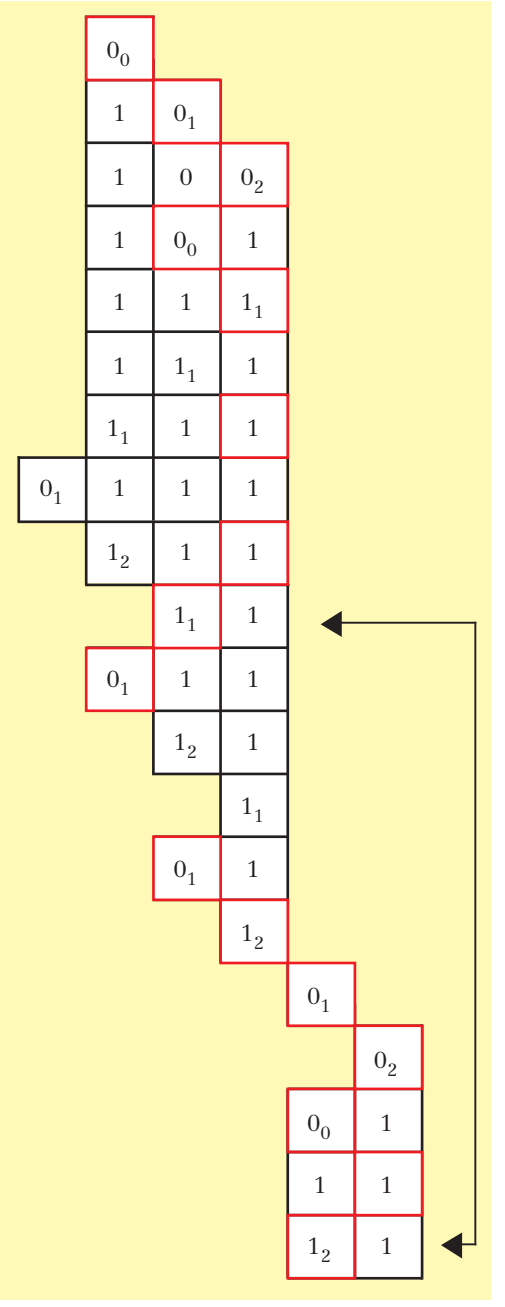


Come si vede, 11 passi bastano per convergere con punteggio 6.

Mostriamo adesso il caso di una macchina che manifesta un comportamento periodico, torna cioè dopo un numero finito di passi su una stessa configurazione, e conseguentemente diverge sullo schermo bianco (ed è estromessa dai due giochi). Essa ha le seguenti istruzioni:

(0, q(0)) → (1, R, q(1)), (1, q(0)) → (1, R, q(3)),
(0, q(1)) → (0, R, q(2)), (1, q(1)) → (1, L, q(1)),
(0, q(2)) → (1, L, q(0)), (1, q(2)) → (0, R, q(1)).

Dunque, l'istruzione di arresto è quella che riguarda la coppia *(1, q(0))*. La computazione sullo schermo bianco avviene allora come adesso descritto:

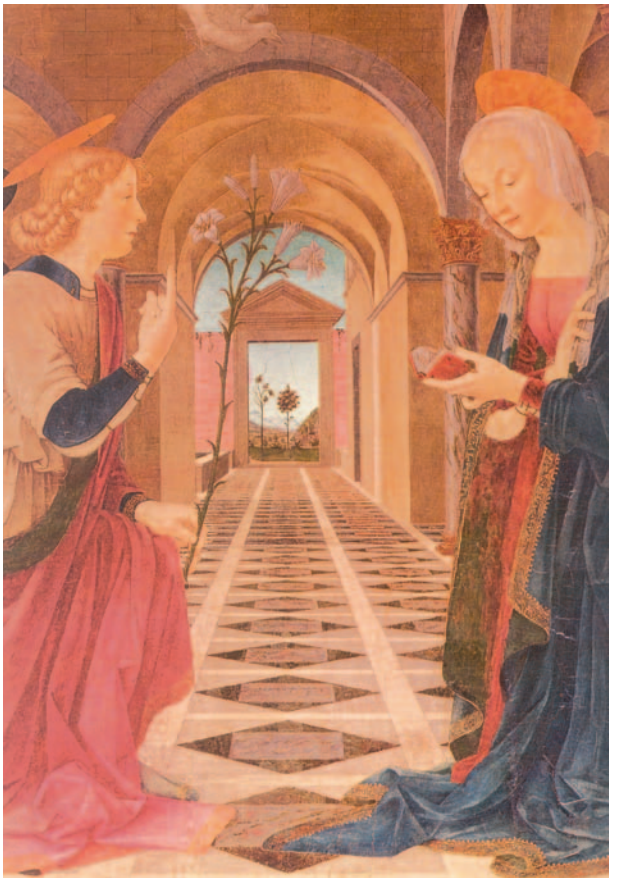


Come si vede, la configurazione dopo 19 spostamenti è la stessa che si ha al nono passo (sia pure su una porzione diversa dello schermo) e questo è sufficiente per dedurre una ripetizione nella computazione e, in definitiva, divergenza. Va detto che, in questi casi "periodici", le cose non sono sempre così semplici: talora il comportamento ripetitivo si manifesta in modo meno lampante e immediato, viene comunque percepito ed implica divergenza e conseguente squalifica.

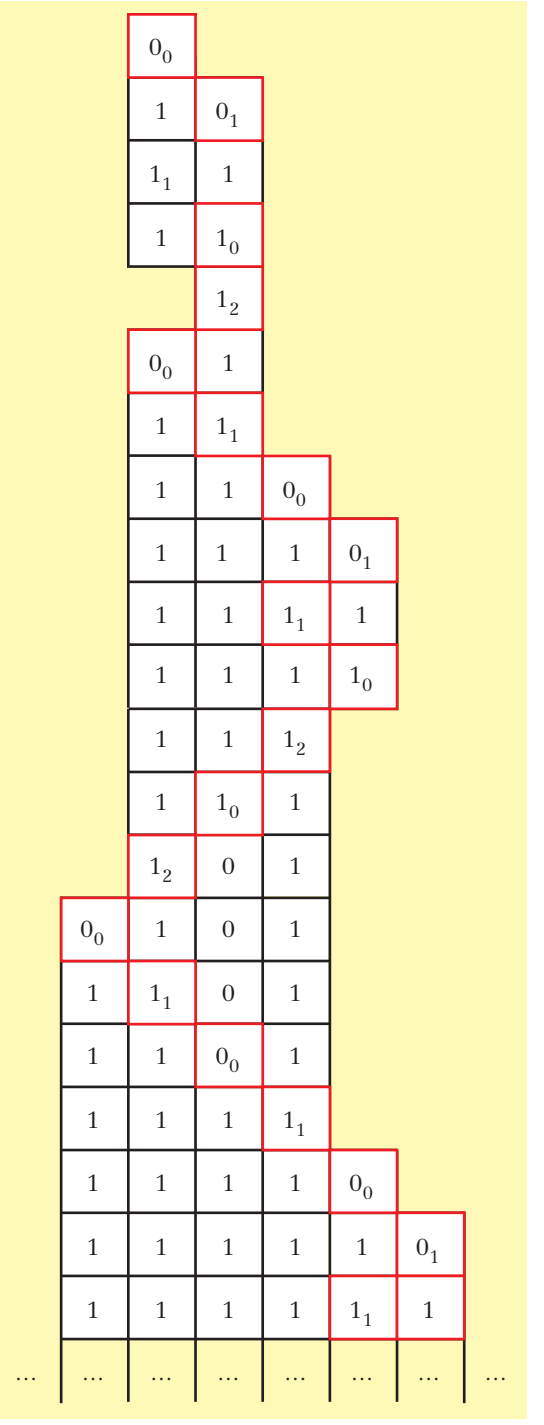
Al termine di questa fase, restano, come già detto, 40 macchine *residue* da esaminare, perché incapaci di convergere o di manifestare ripetizioni entro i primi 21 passi di computazione. Ma basta alzare il livello di controllo, da 21 a 50, per osservare fenomeni di ricorrenza e dunque dedurre la divergenza di ciascuno di questi casi. Proponiamo due esempi, attinti tra i 40 residui, che sono utili per illustrare due tipi di comportamento abbastanza comuni e anticipare così alcuni spunti del caso $n = 4$, che dobbiamo ancora trattare. Consideriamo, dapprima, il caso di una macchina di Turing M a 4 stati con le seguenti istruzioni:

$(0, q(0)) \to (1, R, q(1)), \quad (1, q(0)) \to (0, L, q(2)),$
$(0, q(1)) \to (1, L, q(1)), \quad (1, q(1)) \to (1, R, q(0)),$
$(0, q(2)) \to (1, R, q(3)), \quad (1, q(2)) \to (1, L, q(0)).$

Si noti che l'istruzione di arresto è quella che riguarda la coppia $(0, q(2))$. La computazione di M sullo schermo bianco, descritta nel modo consueto, si sviluppa così per i primi 21 passi:



Annunciazione, Gardner (XV secolo) - Isabella Sewart Gardner Museum, Boston



Già da questi primi spostamenti si percepisce un comportamento ripetitivo che porta M ad allargare la sequenza degli 1 alternativamente a destra e a sinistra del quadro di partenza. La figura che ne deriva ricorda vagamente quella di un albero di Natale (o, più prosaicamente, di un abete). L'analisi della successiva computazione conferma questa im-

pressione. Se ne deduce che M non si arresta e diverge (il che porta a squalificarla dai due giochi di $\Sigma(3)$ e di $S(3)$). Esistono altri esempi di macchine residue che condividono lo stesso comportamento. Esse vengono appunto chiamate *alberi di Natale* (*Xmas trees*, in inglese), proprio a sottolineare l'aspetto esteriore della loro computazione.
Vi sono comunque altri esempi di macchine *residue*, che non corrispondono al modello degli alberi di Natale. Consideriamo infatti le seguenti istruzioni:

$(0, q(0)) \to (1, R, q(1)), \quad (1, q(0)) \to (1, L, q(2)),$
$(0, q(1)) \to (0, L, q(0)), \quad (1, q(1)) \to (0, R, q(1)),$
$(0, q(2)) \to (1, L, q(0)), \quad (1, q(2)) \to (1, R, q(3)).$

L'istruzione di arresto stavolta concerne $(1, q(2))$. La macchina di Turing M che ne deriva esegue la seguente computazione di M sullo schermo bianco (almeno per i primi 21 spostamenti):

Veduta urbana (tarsia lignea), Cristoforo da Lendinara - Museo di villa Guinigi, Lucca

			0_0			
			1	0_1		
			1_0			
		0_2	1			
		0_0	1_2	1_2		
		1	1_1	1		
		1_1	0	1_1		
		1	0	0	0_1	
		1	0	0_0		
		1	0	1	0_1	
		1	0	1_0		
		1	0_2	1		
		1_0	1	1		
	0_2	1	1	1		
0_0	1	1	1	1		
1	1_1	1	1	1		
1	0	1_1	1	1		
1	0	0	1_1			
1	0	0	0	1_1		
1	0	0	0	0	0_1	
1	0	0	0	0_0		
...	...	...	...	...	...	...

Come si vede, la sequenza degli 1 sullo schermo si distribuisce in modo apparentemente irregolare, e sembra difficile intuirne la logica o immaginare con qualche sforzo di fantasia alberi o altre figure familiari. Ma si osservi il rigo 14 della computazione: vi si vedono 3 cifre uguali ad 1, precedute da uno 0 esaminato dalla macchina nello stato $q(2)$, men-

tre il resto dello schermo è completamente bianco. Un esame allargato ai successivi spostamenti porta in effetti prima a congetturare, e poi a provare, che a intervalli regolari M viene a formare sullo schermo stringhe della forma:

…	0	0_2	1	1	1	…	1	1	1	0	…

dove la sequenza centrale di 1 ha lunghezza che aumenta con il numero dei passi computati da M. In questo senso, M ha caratteristiche di *contatore*: dopo un numero prefissato di passi, accumula appunto una sequenza di lunghezza prefissata (e crescente) di 1 consecutivi, preceduti da uno 0 osservato nello stato *q(2)*. Di nuovo, questa osservazione basta per dedurre che M diverge e dunque deve essere squalificata dai nostri giochi. Simili considerazioni permettono di escludere tutte le 40 macchine residue e concludere finalmente $\Sigma(3) = 6$, $S(3) = 13$, come già detto.

Le elucubrazioni appena accennate per $n = 3$ lasciano ben intendere che genere di difficoltà ci possiamo aspettare per:

n = 4.

A questo caso Allen H. Brady aveva lavorato sin dalla sua tesi del 1964, trovando dapprima esempi di macchine a 5 stati che ottenevano punteggio 13 per $\Sigma(4)$ e 107 per $S(4)$ e giungendo così a fissare le limitazioni superiori:

$\Sigma(4) \geq 13, \quad S(4) \geq 107.$

Nel 1975 Brady provò finalmente che le stime 13, 107 coincidono effettivamente con $\Sigma(4)$ e $S(4)$; di questo risultato è data relazione in [1]. È facile immaginare quanto formidabile sia stato il corrispondente lavoro. In effetti, le argomentazioni delle osservazioni *1-5* riducono lo spettro delle macchine di Turing in esame dagli originari 25 miliardi (e oltre) a circa 142 milioni. Ma questo è solo l'inizio delle analisi, cui seguono (come nel caso di 3) l'individuazione di comportamenti chiave ricorrenti tra i 142 milioni di esempi, la formulazione di ipotesi euristiche e il controllo di queste teorie su calcolatori fino ad un numero prefissato di spostamenti. Filtri successivi permettono di escludere quelle macchine che manifestano, entro tale numero limitato di passi, un comportamento ripetitivo (indice di una divergenza complessiva) e di annotare i punteggi delle macchine che, invece, riescono a convergere entro tale ambito. Un programma euristico denominato *BBFILT* (*BB* per *busy beaver*, castoro laborioso, *FILT* per filtro) riesce a completare questa analisi individuando 5.820 macchine *residue*, il cui comportamento resta da verificare.

Di questi 5.820 esempi, 4849 sono di tipo *albero di Natale*, altri 402 corrispondono al modello *contatore*, i restanti 569 hanno caratteristiche anomale e sono catalogati di tipo *sconosciuto*. Ulteriori raffinati perfezionamenti di *BBFILT* riducono a 218 i casi da considerare, mostrando la divergenza di tutti gli altri esempi. Di questi 218 casi, 8 corrispondono ad *alberi di Natale*, 47 a *contatori*, 163 al tipo *sconosciuto*. Di nuovo, si cercano di individuare in questi esempi restanti comportamenti comuni che ne permettano un'analisi generale. La cosa non risulta semplice: in alcuni casi, per esempio, si osserva una computazione ciclica (e dunque si deduce la divergenza) soltanto dopo 422 spostamenti. Si riconoscono comunque talora macchine di Turing che ricordano il tipo dei contatori. Un altro modello di macchina ha meritato il nome fantasioso di *drago che si morde la coda* (*tail eating dragon*, in inglese). La sua computazione avviene così: dopo aver stampato una stringa iniziale (la *coda*, appunto), la macchina inizia a muoversi avanti e indietro come nel caso degli *alberi di Natale* e, contemporaneamente ad ogni oscillazione, mangia un pezzo della coda. Quando quest'ultima è completamente consumata, la macchina stampa una nuova coda, più grande della precedente, e poi ricomincia il processo. Ovviamente, un modello di questo genere, appena riconosciuto, è destinato alla divergenza e può essere eliminato nella ricerca di $\Sigma(4)$ e $S(4)$. In definitiva, tutte le macchine residue vengono esaminate e per nessuna di loro si constata la convergenza. Così, si può concludere:

$\Sigma(4) = 13, \quad S(4) = 107.$

Il lettore sarà curioso di conoscere qualche macchina capace di raggiungere questi punteggi vincenti. In questo caso, ci basta un unico esempio, quello con le istruzioni:

(0, q(0)) → (1, R, q(1)), (1, q(0)) → (1, L, q(1));
(0, q(1)) → (1, L, q(0)), (1, q(1)) → (0, L, q(2));
(0, q(2)) → (1, R, q(4)), (1, q(2)) → (1, L, q(3));
(0, q(3)) → (1, R, q(3)), (1, q(3)) → (0, R, q(0)).

Come si vede, l'istruzione di arresto è quella riguardante *(0, q(2))*. La macchina ammette i punteggi massimi sia per Σ che per S, rispettivamente 13 e 107 appunto. Ma, quanto alla verifica dei 107 passi, la lasciamo volentieri al lettore interessato e non a corto di pazienza.

Le ultime notizie

4 è il massimo naturale per cui oggi, nel 2005, i valori di Σ e di S sono noti con precisione. Per valori interi n superiori a 4, si conoscono attualmente soltanto delle limitazioni in-

feriori per $\Sigma(n)$ e $S(n)$. Per esempio, nel 1964, M. Green aveva determinato delle funzioni computabili di n, che non superano $\Sigma(n)$ per nessun n, ma lo approssimano ragionevolmente, dandone una stima per difetto. Se ne deduce, in particolare, che:

$\Sigma(7) \geq 22.961$,
$\Sigma(8) \geq 8{,}10^{44}$.

Per $n = 5$ e 6, H. Marxen e J. Buntrock hanno trovato limitazioni ancora migliori. La strategia da loro seguita è, per certi versi, ovvia e scontata: si tratta di costruire macchine di Turing rispettivamente a 6 e 7 stati, capaci di ottenere punteggi rilevanti nei giochi di Σ e S. I loro risultati costituiscono limitazioni inferiori dei valori vincenti. In tutti questi casi, tuttavia, resta da sviluppare un'analisi complessiva di tutti i casi possibili, che identifichi in conclusione i punteggi massimi e quindi $\Sigma(n)$ e $S(n)$. La tabella che segue riassume quanto si conosce attualmente di Σ e S; in particolare l'ultima colonna riferisce, riga per riga, chi per primo ha scoperto i valori citati o le relative limitazioni.

n	$\Sigma(n)$	$S(n)$	Fonte
1	1	1	Rado
2	4	6	Rado
3	6	21	Lin-Rado
4	13	107	Brady
5	$\geq$ 4098	$\geq$ 47.176.870	Marxen-Buntrock
6	$> 1{,}29 \cdot 10^{865}$	$> 3 \cdot 10^{1730}$	Marxen-Buntrock

I valori dell'ultima riga, quelli relativi a 6, sono relativamente recenti in quanto furono individuati da H. Marxen e J. Buntrock nel marzo del 2001 e confermati, indipendentemente, da P. Stevens nel luglio del 2001 e da C. Tooth nel giugno del 2002.
La situazione per Σ e S è, ovviamente, passibile di aggiornamenti e sviluppi. La potenza dei moderni calcolatori può far presagire nuove informazioni. Ci sono vari studiosi che a questo problema si appassionano e i loro sforzi sono riferiti da siti internet appositamente dedicati alla funzione di Rado. Alcuni sono citati in appendice, altri ancora possono essere individuati interpellando i motori di ricerca.
D'altra parte, è notevole rilevare quanto poco si conosca ancor oggi di Σ nonostante lo spessore degli strumenti impiegati. Si può davvero dubitare che solo un castoro laborioso ed accanito possa ottenere sostanziali novità. Così, alcuni ricercatori umani si sono impegnati a coltivare nuovi orizzonti ed ambiti, volti ad ammettere macchine i cui alfabeti contengono più di un unico, misero simbolo 1, o addirittura macchine di Turing a più dimensioni, nelle quali le cifre disponibili sono disposte non più in riga, ma in griglie rettangolari, o anche a forma di parallelipipedo. In tutti questi casi, è possibile adattare in modo opportuno la definizione di Σ e avviarne il calcolo nei casi più semplici. L'articolo [2] contiene alcune spunti interessanti a questo proposito. ■

BIBLIOGRAFIA

A. Brady, "The determination of the value of Rado's non-computable function sigma(k) for four-state Turing machines", *Math. Comp.* 40 (1983), pp. 647-665.

A. Brady, "The busy beaver game and the meaning of life", in: *The Universal Turing machine. A Half-century survey*, di R. Herken (ed.), Springer, 1994, pp. 237-254.

S. Lin, T. Rado, *Computer studies of Turing machine problems*, Journal Ass. Comput. Mach. 12 (1965), pp. 196-212.

T. Rado, "On a simple source for non-computable functions", *Proceedings of the Symposium on Mathematical Theory of Automata*, Polytecnic Institute of Brooklyn, 1963, pp. 75-81.

SITI INTERNET

grail.cba.csuohio.edu/~somos/bb.html (M. Somos)

www.drb.insel.de/~heiner/BB/index.html (H. Marxen)

La scuola di Atene (1509-10), Raffaello - Stanza della Segnatura, Vaticano