

# TEORIA DELLA RICORSIVITA'

Piergiorgio Odifreddi

Luglio 1991

I **numeri naturali**  $0, 1, 2, \dots$  sono noti dai primordi dello sviluppo, e oggetto di studio sistematico dai tempi dei Babilonesi, degli Egizi e dei Greci. In un certo senso, essi sono gli oggetti matematici più conosciuti, e vengono usati ovunque e da chiunque: come dice il *Libro dei Morti* egiziano, puoi portarmi un uomo che non sappia contare con le sue dita?

Le operazioni con numeri si chiamano **funzioni** (ci sono ovviamente altri tipi di funzioni, di cui questo volume tratta diffusamente; dovremmo quindi parlare di funzioni *da numeri naturali a numeri naturali*, ma per comodità sottointenderemo sempre questa precisazione).

*La Teoria della Ricorsività è lo studio delle funzioni, ed il suo scopo è di classificarle dal punto di vista della loro difficoltà di calcolo.*

Una prima classificazione, introdotta nella Sezione 1 e analizzata nella Sezione 2, consiste nel separare le **funzioni ricorsive** (cioè, per definizione, quelle calcolabili mediante un computer) dalle altre. Le Sezioni 3 e 4 sono dedicate, rispettivamente, ad una discussione delle potenzialità e delle limitazioni della nozione di ricorsività.

Ulteriori e più raffinate classificazioni si possono ottenere individuando varie altre classi di funzioni, alcune più estese ed altre più ristrette della classe delle funzioni ricorsive. I due approcci sono discussi, rispettivamente, nelle Sezioni 5 e 6.

## 1 I Progenitori

All'idea di funzione ricorsiva si arrivò in due modi indipendenti e complementari, mostrandoci fin dall'inizio come la Teoria della Ricorsività si situi in un'area a cavallo fra matematica ed informatica.

---

<sup>0</sup>Capitolo 9 del Volume XII (*Logica Matematica. Matematica. Informatica*) della **Storia del secolo XX**, Istituto dell'Enciclopedia Italiana fondata da Giovanni Treccani.

## 1.1 Babbage

Abbiamo già definito le funzioni ricorsive come quelle calcolabili mediante un computer. Questa idea risale a Carl Babbage (1791–1871), che nel 1837 concepì (con un anticipo di un secolo) una macchina capace di essere programmata per effettuare tutti i calcoli che un essere umano può teoricamente effettuare. Per passare da una fase speculativa ad una realizzativa, egli si chiese quali operazioni fossero necessarie, e formulò chiaramente il fatto che ci si può restringere a considerare alcune funzioni aritmetiche basilari (somma e prodotto), e alcune operazioni su di esse (ricorsione primitiva e ricerca del minimo, che definiremo tra poco). Egli aveva dunque la nozione di funzione ricorsiva nel senso moderno, e un analogo di quella che chiameremo nella Sezione 3 **Tesi di Church**, cioè l'asserzione che *la nozione di funzione ricorsiva è un equivalente matematico preciso della nozione informale di funzione calcolabile* (così come la nozione di area è un equivalente matematico preciso della nozione informale di misura di una superficie).

Babbage è oggi ben noto come un precursore dei computer, per il suo lavoro di progettazione e di costruzione di vere e proprie macchine calcolatrici. Ma tale lavoro fu riscoperto solo quando i computer erano ormai stati costruiti, e non ebbe quindi un grande influsso tecnologico. L'analisi teorica a cui abbiamo alluso è quindi forse il suo più valido, e tuttora attuale, contributo informatico.

## 1.2 Dedekind

Un'analisi matematica del concetto di numero naturale fu intrapresa e portata a termine da Richard Dedekind (1831–1916), nel 1888. Egli iniziò notando che i numeri naturali sono generati a partire dallo 0, mediante l'operazione  $\mathcal{S}$  (detta *successore*) tale che  $\mathcal{S}(x) = x + 1$ :

$$0 \quad \mathcal{S}(0) = 1 \quad \mathcal{S}(\mathcal{S}(0)) = 2 \quad \dots$$

Poichè i numeri sono tutti della forma 0 o  $\mathcal{S}(x)$  (e non entrambe), Dedekind introdusse il principio di definizione per **ricorsione primitiva**: per definire una funzione su tutti i numeri naturali è sufficiente stabilire il suo valore per 0, e descrivere come si può passare dal valore per  $x$  al valore per  $\mathcal{S}(x)$ . Ad esempio, la funzione  $f(x) = 2^x$  si può definire nel seguente modo:

$$f(0) = 1 \quad f(\mathcal{S}(x)) = 2 \cdot f(x).$$

Dedekind era interessato a trovare proprietà che caratterizzassero la struttura dei numeri naturali in modo univoco, e scoprì che una struttura ordinata con un primo elemento (facente funzione di 0) ed una funzione successore (facente funzione di  $\mathcal{S}$ ) è indistinguibile da (in termine tecnico, *isomorfa a*) i numeri numerali se soddisfa al *Principio del Minimo*, cioè se ogni suo sottoinsieme non vuoto ha un primo elemento. In tal caso è possibile stabilire l'isomorfismo

per ricorsione primitiva (associando a 0 il primo elemento, ed a  $\mathcal{S}(x)$  il successore dell'elemento associato ad  $x$ ), ed il principio del minimo assicura che nessun elemento è lasciato fuori dal processo.

In realtà l'analisi di Dedekind richiede anche le seguenti altre semplici proprietà:

- il primo elemento non è il successore di alcun altro elemento
- ogni elemento che non sia il primo è il successore di un altro elemento
- elementi diversi hanno successori diversi.

Inoltre, per precisione storica, Dedekind non usò il Principio del Minimo, ma una sua forma equivalente detta *Principio di Induzione*. Il Principio del Minimo ha però (come vedremo fra un istante) un'immediata controparte in termini di funzioni, e quindi è più naturale dal nostro punto di vista.

Basandosi sull'analisi di Dedekind, Stephen Kleene (n. 1909) definì nel 1936 le funzioni ricorsive in modo induttivo, partendo dalla funzione costante 0 e dalla funzione successore  $\mathcal{S}$ , e generando nuove funzioni mediante ogni possibile combinazione di ricorsione primitiva e **ricerca del minimo** (che consiste nel trovare il minimo elemento di un insieme non vuoto di numeri naturali).

## 2 Consolidamento

Fra la definizione di Babbage e l'analisi di Dedekind passarono 50 anni, ed altri 50 intercorsero fra quest'ultima e la definizione di Kleene su di essa basata. Con il lavoro di Kleene terminò la fase di consolidamento della Teoria della Ricorsività, che occupò gli anni '20 ed i primi anni '30.

### 2.1 Gli anni '20

Gli anni '20 videro soprattutto un'analisi della potenza della ricorsione primitiva, e la conseguente scoperta dell'interesse della classe delle **funzioni ricorsive primitive** definite mediante tale procedimento. Attraverso i lavori di Paul Bernays (1888–1977), David Hilbert (1862–1943), Rózsa Péter (1905–1977), e Thoralf Skolem (1887–1963) si scoprì che non si ottengono nuove funzioni se si permettono procedimenti di definizione in cui il valore per  $\mathcal{S}(x)$  si ottiene non soltanto dal precedente valore per  $x$ , ma da un numero qualunque di valori per  $0, \dots, x$ .

Che tali procedimenti di definizione non siano soltanto curiosità matematiche, ma sorgano naturalmente dalla considerazione di semplici problemi, è mostrato dal seguente esempio, in cui il valore in un punto è definito mediante i *due* precedenti valori:

$$f(0) = 1 \quad f(1) = 2 \quad f(x+2) = f(x) + f(x+1).$$

Leonardo da Pisa (1180–1250), meglio noto come Fibonacci, introdusse  $f(x)$  nel 1202 come soluzione del seguente problema: determinare quante coppie (monogame) di conigli si generino a partire da una coppia fertile in  $x$  mesi, se si suppone che ogni coppia diventi fertile all'età di due mesi, ed ogni coppia fertile partorisca ogni mese una nuova coppia. Da allora, la funzione  $f(x)$  ha giocato un ruolo importante nella descrizione della crescita degli organismi viventi, specialmente in botanica.

Una definizione primitiva ricorsiva di una funzione ne descrive il comportamento in modo tale da permetterne il calcolo. Ad esempio, nel caso della funzione di Fibonacci si può calcolare il valore  $f(12) = 377$  delle coppie partorite in un anno, calcolando successivamente tutti i valori fino al dodicesimo.

I successi appena descritti fecero dunque sorgere una speranza: visto che anche utilizzando procedimenti di definizione piuttosto complicati non si esce dalla classe delle funzioni ricorsive primitive, forse tale classe contiene tutte (e sole) le funzioni che siano in qualche modo calcolabili, cioè per le quali esista un procedimento che permetta di trovarne i valori in modo meccanico ed in un tempo limitato.

Tali speranze furono distrutte da Wilhelm Ackermann (1896–1962), che scoprì nel 1928 una funzione ‘facilmente’ calcolabile che non è ricorsiva primitiva. L'idea è semplice, ed utilizza un procedimento che risale a Georg Cantor (1845–1918), detto **diagonalizzazione**. L'osservazione basilare è che si possono enumerare le funzioni ricorsive primitive di un argomento in modo effettivo, ad esempio considerando tutte le loro possibili definizioni mediante combinazioni di ricorsioni primitive, e mettendole in fila in *ordine alfabetico*, ciascuna in una pagina di una infinita guida delle funzioni ricorsive primitive. Si ottiene così una lista

$$f_0 \quad f_1 \quad f_2 \quad \dots$$

di tutte le funzioni ricorsive primitive di un argomento. La funzione definita da

$$f(x) = f_x(x) + 1$$

è facilmente calcolabile (mediante la guida), ma non primitiva ricorsiva (perchè una pagina non è un libro).

Per calcolare  $f(x)$  si sfoglia la guida delle funzioni ricorsive primitive fino a pagina  $x$ , si segue la definizione che sta in tale pagina (e che definisce  $f_x$ ), fino a trovare il valore per l'argomento  $x$  (calcolando così  $f_x(x)$ ), e poi si aggiunge 1 a tale valore.

La guida delle funzioni ricorsive primitive non contiene nessuna definizione di  $f$ , perchè la definizione a pagina  $x$  definisce  $f_x$ , ed  $f$  è diversa da  $f_x$  (in quanto ha un valore diverso per l'argomento  $x$ ). Dunque  $f$  non è ricorsiva primitiva.

## 2.2 Gli anni '30

Poichè l'appetito vien mangiando, a questo punto (nei primi anni '30) ci si cominciò a chiedere *quale* fosse allora la classe delle funzioni calcolabili. Natural-

mente, poichè la nozione di calcolabilità era sufficientemente vaga (ed i computers, che avrebbero potuto fornire un modello su cui accordarsi, non erano ancora stati inventati), ciascuno ebbe da dire la sua: Alonzo Church (n. 1903), Kurt Gödel (1906–1978), Emil Post (1897–1954), Alfred Tarski (1902–1983), e Alan Turing (1913–1954) proposero, nel giro di pochi anni, un gran numero di possibili definizioni. Ciascuna era basata su di un particolare aspetto della nozione di funzione calcolabile: di essere definibile mediante un *sistema di equazioni* (del tipo di quelle usate per la funzione di Fibonacci), determinante i valori mediante semplici regole; di ammettere *dimostrazioni matematiche* del fatto che un dato numero è il valore della funzione per dati argomenti; di essere calcolabile mediante particolari *macchine astratte*, capaci di effettuare certe operazioni e certi tipi di decisioni.

Ciascuno di questi modelli di calcolo aveva le sue caratteristiche, e poteva essere proposto come *una* soluzione matematica del problema di caratterizzazione della nozione di calcolabilità. Il problema filosofico era, naturalmente, di fare una scelta convincente. Ma l'imbarazzo non sussistette a lungo, poichè Kleene diede inizio nel 1936 ad una serie di risultati che culminarono nella seguente sorpresa: *tutte le definizioni proposte sono equivalenti, e descrivono sempre la classe delle funzioni ricorsive.*

Il metodo usato nelle dimostrazioni di tali risultati di equivalenza è detto **aritmetizzazione**, e consiste nell'assegnare numeri a oggetti in modo sistematico ed effettivo, e nel tradurre proprietà degli oggetti in proprietà dei loro corrispondenti numeri. La prima parte non era certamente un'idea nuova, poichè già Pitagora di Samo (580–500 A.C.) sosteneva che 'tutto è numero' (anche se egli, ovviamente, non pensava ad una assegnazione sistematica ed effettiva), ed è già stata da noi usata per enumerare le funzioni ricorsive primitive; la seconda parte fu sperimentata secoli fa da Gottfried William Leibniz (1646–1716), nella speranza di poter poi sostituire ragionamenti imprecisi riguardo ad oggetti comuni con rigorosi ragionamenti matematici. Oggi siamo abituati a vedere questo metodo usato dovunque; ad esempio, le targhe automobilistiche sono un'applicazione della prima parte (numeri assegnati a macchine), e il decidere di far circolare le auto a targhe alterne un'applicazione della seconda (una macchina circola un dato giorno a seconda che il suo numero di targa sia pari o dispari). La vera osservazione cruciale fu però opera di Gödel, che notò come l'intero processo di aritmetizzazione si possa effettuare mediante funzioni ricorsive.

Naturalmente, l'equivalenza fra varie definizioni non è ancora una dimostrazione del fatto che la nozione di funzione ricorsiva sia effettivamente una caratterizzazione della nozione di funzione calcolabile, ma certamente prova che la classe delle funzioni ricorsive ha un grande interesse intrinseco.

### 3 I Nuovi Prometei

L'ultimo paragrafo intende mettere in guardia dal trarre precipitose o premature conclusioni dall'equivalenza citata, ma nel 1936 Church e Turing furono meno circospetti, ed enunciarono la cosiddetta **Tesi di Church**, (già anticipata, come abbiamo ricordato, da Babbage): *le funzioni ricorsive sono esattamente le funzioni calcolabili*. Post si spinse fino ad affermare che

se 'funzione ricorsiva' è l'equivalente formale di 'funzione calcolabile', la sua formalizzazione può giocare un ruolo nella storia della matematica secondo soltanto alla formalizzazione del concetto di numero naturale.

In questa sezione illustriamo il significato della tesi di Church, e ne discutiamo brevemente la plausibilità. Possiamo anticipare fin d'ora la conclusione della discussione che segue, dicendo che la Tesi di Church è stata formulata in un eccesso di ottimismo, senza badare troppo alle sue possibili implicazioni. Essa ha certamente aiutato ad attirare l'attenzione sugli aspetti positivi della nozione di ricorsività, ma può oggi essere considerata come facente parte di quella guardia del corpo di falsità di cui, come disse Winston Churchill (1874–1965) che di falsità si intendeva, la verità ha bisogno.

#### 3.1 Dimostrazioni di impossibilità assoluta

Varie volte, nello sviluppo della matematica, si è scoperto che certi problemi sono insolubili mediante dati strumenti. Ad esempio, è noto dai tempi dei Babilonesi che esiste una semplice formula algebrica che permette di calcolare le soluzioni di qualunque equazione di secondo grado

$$ax^2 + bx + c = 0,$$

precisamente:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Scipione del Ferro (1465–1526), Niccolò Fontana (1499–1557) detto Tartaglia, Geronimo Cardano (1501–1576) e Ludovico Ferrari (1522–1565) trovarono formule algebriche che permettono di calcolare le soluzioni di qualunque equazione di terzo o quarto grado. Ma Paolo Ruffini (1765–1822) e Niels Abel (1802–1829) dimostrarono che per nessun  $n > 4$  esistono formule algebriche che permettano di calcolare le soluzioni di qualunque equazione di grado  $n$ .

Similmente (e, come vedremo nella Sezione 4.3, in modo molto facile), è possibile trovare vari problemi che non ammettono soluzioni ricorsive. Usando però la Tesi di Church, da tale impossibilità relativa si può ora dedurre una impossibilità assoluta. Ancora nelle parole di Post:

ciò che è nuovo è che nel caso presente questi strumenti sembrano essere, in effetti, gli unici a disposizione dell'uomo.

L'impatto metafisico della Tesi di Church è dunque enorme, e si può sintetizzare nella seguente affermazione: *la Teoria della Ricorsività fornisce uno strumento per provare matematicamente limitazioni delle possibilità del pensiero umano.*

### 3.2 Limitazioni e potenza dei computers

Fra le varie definizioni equivalenti di funzione ricorsiva abbiamo citato la nozione di calcolabilità mediante macchina astratte. Tale approccio è dovuto a Turing e Post, e le cosiddette **macchine di Turing** si possono oggi facilmente descrivere: esse sono i computers, astratti dalle loro limitazioni fisiche (come la possibilità di rotture o di cattivo funzionamento, o la limitatezza di memoria). Qualunque computer (in tale senso astratto) è in grado di calcolare ogni funzione ricorsiva di cui gli sia fornito un programma. La Tesi di Church asserisce che le sole funzioni per cui esistono programmi sono le funzioni ricorsive, e dunque che *abbiamo raggiunto i limiti della potenza di calcolo meccanico*: sarà possibile in futuro migliorare l'efficienza dei computers (rendendoli più veloci, comprimendone le dimensioni, abbassandone i costi), ma non la loro potenza assoluta. In particolare, così come già per il pensiero umano, la Teoria della Ricorsività fornisce uno strumento per provare matematicamente limitazioni delle possibilità dei computers.

Il riferimento ai computers non è una concessione all'ultima moda tecnologica. E non è neppure una contraddizione l'aver detto che i computers non esistevano al tempo della formulazione della Tesi di Church, ma che la definizione di funzione ricorsiva proposta da Turing e Post è equivalente alla calcolabilità mediante computers. La realtà è che, in questo caso, l'analisi teorica precedette la realizzazione tecnologica, e fu una parte influente nei due progetti che portarono alla costruzione dei primi computers: l'ENIAC (*Electronic Numerical Integrator and Calculator*, Integratore e Calcolatore Numerico Elettronico) diretto negli Stati Uniti da John Von Neumann (1903–1957), e l'ACE (*Automatic Computing Engine*, Macchina Calcolatrice Automatica) diretto in Gran Bretagna da Turing stesso, entrambi negli anni a cavallo del 1950.

La Tesi di Church non ha però soltanto applicazioni limitative delle possibilità dei computers. Usando la caratterizzazione delle funzioni ricorsive in termini di calcolabilità mediante computers, da essa si ottiene l'altra faccia della medaglia: *le funzioni calcolabili sono calcolabili mediante computers*. Questa non è più un'asserzione di limitazione, bensì di potenza dei computers. Per comprenderla appieno, dobbiamo analizzarne più da vicino che cosa significhi l'aggettivo 'calcolabile'.

### 3.3 Il funzionamento dell'universo

Una funzione è calcolabile da un punto di vista *fisico* se esiste un meccanismo che fornisce i suoi valori. Naturalmente, l'affermazione che ogni tale funzione sia calcolabile mediante un computer richiede, per non essere circolare, che si intenda 'meccanismo' in un senso sufficientemente generale (e non, ad esempio, soltanto come un computer). Per poter tenere conto non solo di ciò che la tecnologia ci ha offerto sino ad oggi, ma anche di ciò che potrà offrire in futuro, ed allo stesso tempo non essere antiscientifici (o, più precisamente, antifisici), possiamo considerare come 'meccanismo' ogni sistema fisico che si evolva secondo le leggi fisiche oggi note. L'idea che un qualunque sistema fisico in evoluzione possa 'calcolare' è ben nota: ad esempio, un termometro fornisce dei dati numerici sulla temperatura attraverso le modificazioni di lunghezza di una colonnina di mercurio. Ma allora la dimessa affermazione che ogni funzione calcolabile è calcolabile mediante un computer diventa la meno ovvia affermazione che *ogni sistema fisico in evoluzione si può simulare mediante un computer*. Un pò meno precisamente, ma in modo ancora più stimolante, si potrebbe dire che il mondo stesso è un gigantesco computer.

Da un punto di vista storico non si dovrebbe prestare molta fede a tali affermazioni, che d'altra parte non sono poi così nuove: in vari momenti cruciali della storia della tecnologia l'invenzione di una nuova 'macchina' portò con sé l'affermazione che il mondo stesso era una tale macchina. Ricordiamo, ad esempio, l'orologio meccanico nel secolo XVII (su cui si basò la visione meccanicista dell'universo), e la macchina a vapore nel secolo XIX (in seguito alla quale l'universo fu visto come un sistema termodinamico). Il computer è, da questo punto di vista, soltanto un analogo per il secolo XX.

L'evidenza a disposizione non è poi così favorevole a questa versione fisica della Tesi di Church. Già al livello di fenomeni descrivibili mediante le leggi della meccanica, incominciano a sussistere dubbi *pratici* sulla sua accuratezza; ad esempio, non si conoscono metodi effettivi per descrivere il comportamento di tre (per non parlare di un numero qualunque di) corpi che si muovano sotto l'influsso di forze gravitazionali newtoniane (il cosiddetto *problema dei tre corpi*). Ma quando si entra nell'ambito della meccanica quantistica, che regola i fenomeni atomici, il problema diventa *teorico*, perchè le leggi su cui essa oggi si basa non sono deterministiche, e non si sa se esistano leggi deterministiche da cui quelle non deterministiche si possano derivare (il cosiddetto *problema delle variabili nascoste*, che sembra essere stato risolto recentemente in modo negativo, sia dal punto di vista teorico, mediante il cosiddetto *teorema di Bell*, che dal punto di vista sperimentale).

### 3.4 La natura del cervello e del pensiero

Una funzione è calcolabile da un punto di vista *umano* se si possono ottenere i suoi valori mediante processi di calcolo cerebrali o mentali. La distinzione

è opportuna, perchè la tradizione filosofica occidentale distingue tra cervello e mente.

Il dibattito moderno sull'esistenza della mente cominciò con René Descartes (1596–1650). Egli propose una visione del mondo puramente meccanicistica, eccezion fatta per l'esistenza di Dio e dell'anima. A favore di quest'ultima Descartes portò come argomenti l'autocoscienza e la sofisticazione del linguaggio umano, che egli considerò come caratteristiche impossibili per animali o macchine.

Thomas Hobbes (1588–1679) propose un'opinione radicalmente opposta. Da un lato, egli considerò il ragionamento come una manipolazione meccanica di nomi fungenti da simboli per i pensieri. D'altro lato, egli ritenne la prima macchina calcolatrice, prodotta da Blaise Pascal (1623–1662) nel 1645 e capace di fare somme (!), una sufficiente refutazione delle affermazioni limitative della capacità delle macchine.

Da un punto di vista moderno, non si può certo dire che gli argomenti di Descartes o Hobbes a favore o contro l'esistenza della mente o la plausibilità del meccanicismo fossero conclusivi. Essi ci appaiono oggi piuttosto rudimentali, ma lo sviluppo scientifico e tecnologico ha portato acqua ad entrambi i mulini.

Da un lato, a supporto di Hobbes, molta parte della matematica è stata effettivamente meccanizzata (in certi casi, come per la cosiddetta 'logica dei predicati', in modo dimostrabilmente completo), e la qualità delle macchine calcolatrici è cresciuta in modo impressionante (al punto che, appunto, la Tesi di Church esprime la fiducia che si siano ormai raggiunti i limiti del possibile).

D'altro lato, a supporto di Descartes, i teoremi di incompletezza di Gödel (considerati nella Sezione 4.4 e, in maggior dettaglio, nel Capitolo 10) indicano sostanziali limiti al processo di meccanizzazione, ad esempio provando che nessun computer può fornire risposte corrette ad ogni domanda riguardo ai numeri.

Considerare cervello e mente come entità separate ci porta dunque a discutere due conseguenze dell'affermazione che ogni funzione umanamente calcolabile è calcolabile da un computer.

Per quanto riguarda il cervello, essa si traduce nell'affermazione che *le attività cerebrali di natura logica o deduttiva si possono simulare mediante un computer*. Norbert Wiener (1894–1964) e Turing hanno addirittura lasciato cadere ogni qualificazione, dando così inizio al sogno dell'Intelligenza Artificiale, nella forma: *le attività cerebrali si possono simulare mediante i computers*.

Affermazioni globali di questa natura non sono nuove, e storicamente si è sempre stati tentati di assimilare il cervello (così come abbiamo già visto per il mondo intero) alla più sofisticata macchina disponibile: nel secolo XVII Descartes lo considerò un sistema idraulico che permette il flusso periodico di spiriti vitali da un bacino centrale ai muscoli; e nel secolo XIX Karl Pearson (1857–1936) lo descrisse come un sistema telefonico consistente di fili fissi e collegamenti mobili (per non cadere nell'eccesso opposto, di ridicolizzare tali tentativi di comprensione, si ricordi che tale modello fu utile per la comprensione del riflesso spinale). Il computer è, da questo punto di vista, soltanto un recente

analogo per il secolo XX.

Per quanto riguarda la mente, l'affermazione che ogni funzione umanamente calcolabile è calcolabile mediante un computer si traduce in: *le attività mentali matematiche si possono simulare mediante un computer*. Questa formulazione della tesi di Church si riferisce alla nozione di attività mentale matematica, che è stata oggetto di studi approfonditi da parte di Luitzen Brouwer (1881–1967) sotto l'etichetta di *intuizionismo* (discusso nel Capitolo 13).

Come già per il caso della calcolabilità fisica, l'evidenza a disposizione non è certo favorevole alla versione umana della tesi di Church. Da un lato, praticamente niente è noto sulle relazioni tra le funzioni intuizioniste e quelle ricorsive. D'altro lato, le differenze fra cervello e computer sono grandi: il cervello è un organo elettrochimico con un gran numero di connessioni, che opera massicciamente con azioni parallele e globali (olistiche), a bassa velocità e basso costo energetico, capace di generare in continuazione nuovi elementi e nuove connessioni; il computer è invece un sistema elettronico a connessioni fisse, operante quasi solo sequenzialmente e localmente, e ad alta velocità.

Soltanto ad un livello estremamente rozzo (benchè con conseguenze di grande utilità pratica) si può pensare di creare un parallelo fra cervello e macchine elettroniche. Nel 1943 Warren McCulloch (1898–1969) e Walter Pitts cercarono una prima approssimazione, facendo alcune ipotesi semplificative sulla struttura e il funzionamento dei neuroni: che essi siano componenti del sistema nervoso con risposte di tipo 'tutto o niente' (e non, come in pratica, a risposta graduale), aventi un livello di soglia fisso (e non, come in pratica, variabile) che determina la presenza o assenza di risposta, ed entranti in azione in modo sincronizzato ad intervalli regolari (e non, come in pratica, in istanti indipendenti l'uno dagli altri), quando la somma algebrica degli impulsi dei neuroni adiacenti raggiunge la soglia. McCulloch e Pitts provarono che un tale sistema si comporta realmente come una macchina (che essi chiamarono **automa finito**), nel senso che ogni sistema di stimoli produce un'unica e determinata risposta.

Nonostante il fatto che tali successi iniziali siano stati sostanzialmente migliorati negli anni seguenti, mostrando in particolare come ipotesi radicalmente meno restrittive di quelle precedenti producono ancora un comportamento deterministico, e dunque un modello di macchina che approssima meglio il comportamento reale del cervello, si è ancora ben lungi dal sogno fantascientifico di sintetizzare un vero cervello.

In ogni caso, anche una macchina che simulasse completamente le funzioni cerebrali non proverebbe automaticamente nè che il cervello è una macchina, nè che la mente non esiste. La prima asserzione richiederebbe un'analisi del reale comportamento del cervello fisico, e non solo una sua simulazione. La seconda è senza speranza: si potrebbe arrivare a dimostrare che la mente non è necessaria per i processi mentali, simulandoli completamente, ma questo non proverebbe che essa non esiste. D'altra parte, è invece possibile (anche se, almeno per ora, non realistico) sperare di dimostrare, in via di principio, che la mente esiste, provando che la struttura fisica del sistema nervoso ha limitazioni potenziali

che sono superate dall'attività mentale reale.

## 4 Sviluppo

Nella sezione precedente abbiamo isolato alcuni temi legati principalmente all'estremismo filosofico (malattia infantile della scienza) che ruota attorno alla Tesi di Church. Gli aspetti filosofici divennero meno prominenti con la crescita tecnologica, di cui trattiamo ora alcuni successi.

### 4.1 Computers e linguaggi di programmazione

Gli automi finiti trovarono un'applicazione immediata e fondamentale nella nascente industria dei computers, nel modo seguente. Turing aveva caratterizzato la nozione di calcolabilità supponendo di avere una scatola nera che fosse in grado di effettuare alcune operazioni elementari. Per l'effettiva costruzione di un computer restava da capire come una tale scatola nera fosse realizzabile, e il lavoro di McCulloch e Pitts fornì la chiave: un automa finito era sufficiente. In pratica un tale automa può essere sintetizzato mediante fili elettrici, in cui le connessioni prendono il posto dei neuroni, e il passaggio o meno di una corrente elettrica prende il posto della presenza o assenza di una risposta. Gli automi finiti, nati come modelli semplificati del cervello *umano*, diventano dunque cervelli *elettronici* per le macchine di Turing, rendendone possibile la costruzione. E così fu in pratica: al lavoro di McCulloch e Pitts si riferirono esplicitamente sia Von Neumann che Turing, nei loro già citati rapporti ENIAC ed ACE.

Come è noto, i computers calcolano funzioni mediante programmi scritti in linguaggi di programmazione, di cui esiste una gran varietà. La Teoria della Ricorsività fornisce da un lato una spiegazione teorica di tale varietà, e dall'altro una guida euristica per la ricerca di nuovi linguaggi: poichè i computers calcolano tutte e sole le funzioni ricorsive, *ogni definizione equivalente di ricorsività descrive un approccio alternativo alla calcolabilità attraverso computers, e dunque un tipo di linguaggio di programmazione* (senza, ovviamente, specificarne i dettagli, e quindi permettendone diverse realizzazioni pratiche).

La dimostrazione di equivalenza fra un dato approccio e la calcolabilità mediante computers consiste nel tradurre il metodo di calcolo di una generica funzione implicito nel dato approccio in una serie di istruzioni eseguibili direttamente dal computer (il cosiddetto *linguaggio macchina*). In termini moderni, tale dimostrazione consiste nella costruzione di un cosiddetto **compilatore**. Essa fu effettuata per la prima volta da Turing e Kleene, mediante il metodo di aritmetizzazione già citato.

## 4.2 Programmi contro funzioni

Una proprietà delle funzioni implicitamente adottata nella trattazione da noi fatta finora è che esse attribuiscono un valore ad ogni argomento. In termine tecnico, abbiamo finora considerato **funzioni totali**. In realtà, a prima vista sembra necessario che così debba essere: già Aristotele (384–322 A.C.) sosteneva (nella *Metafisica*) che non ci possono essere leggi logiche precise per proposizioni che parlino di oggetti non completamente definiti, e la sua opinione è stata condivisa fino all'inizio del nostro secolo.

Con l'avvento dei computers, però, le funzioni hanno ceduto il loro ruolo di oggetti privilegiati ai programmi o, in termini più astratti, ai metodi di calcolo. Ovviamente un programma definisce una funzione in modo indiretto, dicendo come calcolarne i valori, ma spesso non è semplice descrivere tale funzione direttamente. Ad esempio, ci vollero 500 anni perchè qualcuno, più precisamente Abraham De Moivre (1667–1754) nel 1730, trovasse una descrizione diretta della funzione di Fibonacci citata in precedenza: per curiosità, essa è la funzione

$$f(x) = \frac{1}{\sqrt{5}} \left[ \left( \frac{1 + \sqrt{5}}{2} \right)^{x+2} - \left( \frac{1 - \sqrt{5}}{2} \right)^{x+2} \right],$$

in cui appare inaspettatamente (almeno per quanto riguarda la connessione con la riproduzione dei conigli) l'affascinante *sezione aurea o divina proporzione*  $\frac{1+\sqrt{5}}{2}$ , un invadente rapporto che si intruffola nei più svariati campi: la geometria (ad esempio, fra le lunghezze delle diagonali di un pentagono regolare, che formano una stella a cinque punte, ed i suoi lati<sup>1</sup>), l'anatomia (ad esempio, fra le lunghezze di falangi consecutive delle dita della mano), l'architettura (ad esempio, fra la lunghezza e l'altezza delle facciate laterali del Partenone), e così via.

Un'altra conseguenza, dal nostro punto di vista più importante, del fatto che si definiscano funzioni indirettamente mediante programmi è che un metodo di calcolo perfettamente definito non produce necessariamente un valore: ad esempio, la formula per la risoluzione delle equazioni di secondo grado calcola sì le soluzioni, ma non dà nessun risultato se  $b^2 - 4ac < 0$ .

La difficoltà dovuta al fatto che certi metodi di calcolo non producono valori di un certo tipo si può ovviamente aggirare con un artificio linguistico, dicendo che in realtà vengono prodotti valori di un tipo diverso. Ad esempio, nel caso delle equazioni di secondo grado si dice che si ottengono soluzioni *immaginarie*. Ma, come la parola scelta dai matematici dimostra, questo non cambia la situazione: i valori del nuovo tipo hanno proprietà sostanzialmente differenti da quelli del vecchio tipo (detti, con evidente sottolineatura, *reali*).

---

<sup>1</sup>Tale stella fu usata come simbolo dai Pitagorici alle Brigate Rosse, e la si ritrova nella tradizione ebraica come 'sigillo di Salomone', in Goethe come 'piede di strega' che impedisce a Mefistofele di uscire dallo studio di Faust, e nelle stelle della bandiera degli Stati Uniti.

Nel caso generale dei programmi, che descrivono semplicemente certi procedimenti di calcolo, non ci si può dunque aspettare che essi producano sempre un risultato. Ad esempio, se si cerca di calcolare il valore  $f(0)$  usando il programma

$$f(x) = 2 \cdot f(\mathcal{S}(x))$$

si entra in un circolo vizioso: per calcolare  $f(0)$  si deve conoscere  $f(1)$ , per calcolare  $f(1)$  si deve conoscere  $f(2)$ , e così via. Si noti, come curiosità, che c'è una (ed una sola) funzione totale che soddisfa la definizione per ogni  $x$  (precisamente, la funzione i cui valori sono tutti 0), ma i suoi valori non si possono calcolare usando il programma.

Se si vuole dare un significato ai programmi in termini di funzioni bisogna quindi considerare **funzioni parziali**, che possono non avere valori (in termine tecnico: essere *indefinite*) per certi argomenti (nei casi estremi, tutti o nessuno; le funzioni totali sono dunque particolari funzioni parziali). Anche qui, come sopra, si può usare un artificio linguistico, e dire che una funzione indefinita per un certo argomento ha in realtà un valore infinito  $\infty$ . Ma le proprietà di  $\infty$  non sono le stesse di quelle dei numeri finiti: ad esempio,  $\infty = \infty + 1$  (il che ci servirà alla fine di questa sezione).

Per distinguerle dalle funzioni totali, indicheremo le funzioni parziali con lettere greche, ad esempio  $\varphi$ . Inoltre, useremo il simbolo speciale  $\simeq$  per l'uguaglianza tra funzioni parziali, e

$$\varphi(x) \simeq \psi(x)$$

significherà che  $\varphi(x)$  e  $\psi(x)$  sono entrambe indefinite o entrambe definite, e che nell'ultimo caso esse hanno lo stesso valore.

Nel 1938 Kleene definì le **funzioni parziali ricorsive**, con una semplice modifica della definizione di funzione ricorsiva della Sezione 1.2: egli permise la ricerca del minimo elemento di un insieme, senza più richiedere che l'insieme fosse non vuoto (se è vuoto non esiste nessun elemento, e non si troverà dunque nessun minimo). Anche le altre definizioni delle funzioni ricorsive si possono adattare al caso delle funzioni parziali e, analogamente a quanto visto nella Sezione 2.2 per le funzioni totali, si dimostra che le varie definizioni sono tutte equivalenti. In particolare, così come le funzioni ricorsive totali erano esattamente le funzioni *totali* calcolabili con computers, *le funzioni ricorsive parziali sono esattamente le funzioni calcolabili con computers*.

Esattamente come avevamo già fatto per le funzioni ricorsive primitive nella Sezione 2.1, le funzioni ricorsive parziali di un argomento si possono enumerare in modo effettivo, ad esempio considerando tutti i programmi di un linguaggio di programmazione in ordine alfabetico. Si ottiene così una lista

$$\varphi_0 \quad \varphi_1 \quad \varphi_2 \quad \cdots$$

di tutte le funzioni ricorsive parziali di un argomento. Il fatto interessante è che tale lista è essa stessa ricorsiva parziale, cioè che esiste un singolo programma di

due argomenti capace di fare da solo ciò che tutti i programmi di un solo argomento fanno. In una parola, si ha il cosiddetto **Teorema di Enumerazione**: la funzione

$$\varphi(x, z) \simeq \varphi_x(z)$$

è ricorsiva parziale.

Per calcolare  $\varphi(x, z)$ , anzitutto si scrivono due sottoprogrammi: uno che enumera sistematicamente i programmi per funzioni di un solo argomento in ordine alfabetico, l'altro che simula un dato programma su un dato argomento. Si usano poi anzitutto il primo sottoprogramma per trovare l' $x$ -esimo programma per funzioni di un solo argomento (esso definisce  $\varphi_x$ ), e poi il secondo sottoprogramma per simulare l' $x$ -esimo programma sull'input  $z$  (calcolando così  $\varphi_x(z)$ ).

Tutto ciò puzza di bruciato, perchè se  $\varphi_x(z)$  è ricorsiva parziale così sono  $\varphi_x(x)$  (che si ottiene come caso particolare, quando  $z = x$ ) e  $\varphi_x(x) + 1$ . Ma quest'ultima non dovrebbe essere ricorsiva parziale, per la seconda parte dell'argomento di Ackermann nella Sezione 2.1 (la prima parte di tale argomento in pratica prova il Teorema di Enumerazione). In realtà, tale parte non può però essere riprodotta qui, a causa della parzialità delle funzioni coinvolte: è perfettamente possibile che

$$\varphi_x(x) \simeq \varphi_x(x) + 1,$$

se entrambi i lati sono indefiniti (perchè  $\infty = \infty + 1$ ), e non si arriva dunque ad alcuna contraddizione.

Il Teorema di Enumerazione è una importante proprietà della classe delle funzioni ricorsive parziali, e prova come tale classe abbia la capacità di 'descrivere sè stessa'. Nella Sezione 4.5 esamineremo più da vicino alcune implicazioni di tale capacità autoreferenziale delle funzioni ricorsive parziali (e dei programmi che le calcolano).

Dal punto di vista informatico, il Teorema di Enumerazione spiega come sia stato possibile passare da macchine per *specifiche* funzioni ricorsive (come le calcolatrici tascabili) a macchine universali per *tutte* le funzioni ricorsive (come i computers): queste ultime sono semplicemente macchine per la specifica funzione  $\varphi(x, z)$ , e quindi calcolano indirettamente ogni  $\varphi_x$ , quando si fissi  $x$ .

### 4.3 Il Problema della Fermata

Avendo introdotto la nozione di funzione ricorsiva parziale, sorge spontaneo chiedersi se sia possibile accorgersi (ad esempio in modo ricorsivo) se un programma calcola una funzione totale. Ma è facile notare che così non è: se ci fosse un programma che decide, dato  $x$ , se  $\varphi_x$  è totale oppure no, dalla lista

ricorsiva parziale

$$\varphi_0 \quad \varphi_1 \quad \varphi_2 \quad \dots$$

di tutte le funzioni ricorsive parziali si potrebbe estrarre una lista ricorsiva

$$f_0 \quad f_1 \quad f_2 \quad \dots$$

di tutte le funzioni ricorsive totali, semplicemente eliminando dalla precedente, una ad una, le funzioni che risultano non essere totali. Ma questa volta a tale lista si potrebbe sì applicare la seconda parte dell'argomento di Ackermann nella Sezione 2.1, e trovare che la funzione

$$f(x) = f_x(x) + 1$$

è una funzione ricorsiva totale che non sta nella lista. Questo prova che *il problema di decidere se un programma calcola una funzione totale non è ricorsivo*, e mostra come sia stata proprio l'introduzione della nozione di funzione parziale ad aver permesso di ottenere il Teorema di Enumerazione.

Il problema della totalità di una funzione è solo uno dei vari problemi senza soluzioni ricorsive a cui abbiamo alluso nella Sezione 3.1, e che la Tesi di Church dunque interpreta come umanamente impossibili da risolvere. Il più famoso di tali esempi è il cosiddetto **Problema della Fermata**: decidere se una data funzione ricorsiva parziale è definita per un dato argomento  $x$ , equivalentemente, se un computer si ferma (e ottiene dunque un valore) quando calcola seguendo le istruzioni di un dato programma per un dato argomento.

Un caso particolare, che già non ammette una soluzione ricorsiva, è quello di decidere se  $\varphi_x$  è definita per l'argomento  $x$ . Se ci fosse un programma che decide tale questione, sarebbe possibile usarlo come sottoprogramma per definire il seguente programma:

$$f(x) = \begin{cases} \varphi_x(x) + 1 & \text{se } \varphi_x(x) \text{ è definita} \\ 0 & \text{se } \varphi_x(x) \text{ è indefinita.} \end{cases}$$

Ma  $f$  sarebbe allora ricorsiva, perchè calcolata da un programma, e dunque dovrebbe stare nella lista di tutte le funzioni ricorsive parziali, cioè dovrebbe essere  $\varphi_x$  per qualche  $x$ . E questo è impossibile, nel primo caso perchè  $f(x)$  e  $\varphi_x(x)$  hanno valori diversi, nel secondo perchè  $f(x)$  è definita mentre  $\varphi_x(x)$  non lo è.

L'insolubilità del Problema della Fermata si può riformulare come segue: *se un programma che fornisce risposte a domande riguardanti la fermata di dati programmi per dati argomenti non mente mai, esso non può fornire tutte le risposte*. Poichè in genere è facile rispondere affermativamente a tali domande (semplicemente simulando il dato programma sui dati argomenti, ed aspettando fino a che si ottenga un valore), la limitazione si ha nel caso di risposta negativa (nel qual caso la simulazione precedente non giunge mai a termine).

## 4.4 Il Teorema di Gödel

La formulazione precedente del Problema della Fermata mostra una grave limitazione delle capacità assolute dei programmi. Ma ciò appare ancora più grave quando si considerino i programmi in due modi complementari: da un lato, come abbiamo già notato più volte, i programmi si possono codificare mediante numeri, e dunque affermazioni sul loro comportamento sono, mediante aritmetizzazione, affermazioni di proprietà dei numeri; d'altro lato, l'attività matematica che consiste nel derivare teoremi da assiomi mediante regole si può essa stessa codificare mediante programmi. Possiamo allora riformulare ulteriormente l'insolubilità del Problema della Fermata, ed ottenere una versione del cosiddetto **Teorema di Gödel** (trattato più estesamente nel Capitolo 10): *un sistema di assiomi e regole che sia codificabile mediante un programma e che non menta mai non può fornire risposte a ogni domanda riguardo i numeri*. In questa formulazione è in gioco l'intera attività matematica formulata attraverso assiomi e regole.

Ai primordi della matematica Babilonesi ed Egizi enunciarono vari interessanti risultati, ma senza dire perchè essi si dovessero ritenere corretti. I Greci introdussero il concetto di dimostrazione, che consiste nel ridurre col ragionamento affermazioni complesse ad altre più semplici, e così via fino ad affermazioni ritenute evidenti, dette *assiomi*. Euclide di Alessandria (300 A.C. circa) tentò di stabilire, negli *Elementi*, un tale sistema assiomatico su cui tutta la matematica del suo tempo si potesse fondare.

Un sistema assiomatico è un primo passo verso la meccanizzazione dell'attività matematica, ma presuppone implicitamente un metodo di ragionamento (o deduzione). Questo può funzionare, e funzionò, fino a quando non sorsero problemi di meccanizzazione. Ma non appena si pensi di usare macchine per dimostrare teoremi, o anche solo verificare dimostrazioni, anche il ragionamento deve essere completamente automatizzato. Il primo a concepire tale possibile uso delle macchine fu Leibniz, che sperava che un giorno ogni disputa si sarebbe potuta risolvere mediante calcoli su appropriate macchine (le sue idee sull'aritmetizzazione, citate nella Sezione 2.2, erano collegate a questo tipo di applicazione). Nel 1847 George Boole (1815–1864) e Augustus De Morgan (1806–1871) iniziarono dunque uno studio delle leggi del pensiero che fu portato a termine nel 1879 da Gottlob Frege (1848–1925), il quale espose un sistema di assiomi e regole (risultato poi completo) per il ragionamento matematico (la cosiddetta *logica dei predicati*).

Alfred North Whitehead (1861–1947) e Bertrand Russell (1872–1970) pubblicarono, fra il 1910 e il 1913, i *Principia Mathematica*. In tale opera essi cercarono di ripetere il successo di Euclide, e di formulare un sistema di assiomi e regole su cui tutta la matematica del loro tempo si potesse fondare. In realtà, essi furono ben più ambiziosi, e sperarono di aver isolato un sistema che rendesse conto non solo del momento contingente, ma anche dell'intera matematica futura.

Il successo di Euclide durò due millenni, ed il suo libro era ancora usato all'inizio del secolo in varie scuole. Il successo di Whitehead e Russell durò appena due decenni: il Teorema di Gödel provò che il loro sistema (che è facilmente codificabile mediante un programma) era ben lungi dall'essere completo, e non poteva provare neppure tutti i fatti veri sui numeri. Se ciò poteva essere loro di consolazione, il Teorema di Gödel asserì anche che la colpa non era di Whitehead e Russell: nessun sistema del genere può esistere, e la verità matematica non può essere compresa nè in sistemi assiomatici, nè in programmi di computers.

## 4.5 Riproduzione e autoriproduzione

Come abbiamo visto, i programmi dei computers definiscono funzioni mediante metodi di calcolo. Poichè però tali metodi riducono il calcolo di dati valori di una funzione al calcolo di altri valori della stessa funzione e mediante lo stesso metodo, c'è in essi qualcosa di circolare. Ad esempio, il programma

$$\varphi(x) \simeq \varphi(x)$$

non dà nessuna indicazione pratica su come si possa calcolare un qualunque valore di  $\varphi$ , ed inoltre la condizione espressa da tale programma è soddisfatta da *ogni* funzione, parziale o totale che sia. In che senso dunque *una* funzione è definita da tale programma?

Per avere un'idea di come si possono affrontare tali problemi, ricordiamo che anche le equazioni possono sembrare a prima vista circolari. Ad esempio,

$$x = 1 + \frac{1}{x}$$

definisce  $x$  in termini di sè stesso.

Un modo diretto per calcolare il valore di  $x$  consiste nel prendere seriamente l'equazione e trarne le conseguenze, continuando a sostituire la parte sinistra con la parte destra:

$$x = 1 + \frac{1}{x} = 1 + \frac{1}{1 + \frac{1}{x}} = \dots = 1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{\dots}}}$$

Si ottiene in tal modo una cosiddetta *frazione continua*, del tipo di quelle considerate per la prima volta da Pietro Antonio Cataldi (1548–1626) per esprimere radici quadrate: benchè oggetti infiniti, esse possono essere troncate a piacere, e forniscono così approssimazioni del valore finale.

Un modo indiretto per trovare il valore di  $x$  consiste nel trasformare, mediante semplici passaggi algebrici, l'equazione data in qualcosa di più familiare:

$$x^2 - x - 1 = 0.$$

In questo caso si è fortunati e si ottiene un'equazione di secondo grado, per la quale si ha la formula risolutiva citata nella Sezione 3.1, che fornisce

$$x = \frac{1 \pm \sqrt{5}}{2},$$

cioè i due numeri usati per la descrizione diretta della funzione di Fibonacci citata nella Sezione 4.2. In particolare, la frazione continua trovata sopra è dunque una descrizione della sezione aurea.

Ritornando ai programmi, nel 1952 Kleene stabilì l'analogo del primo metodo mediante il cosiddetto **Teorema di Ricorsione**. Esso dice semplicemente che, nonostante il fatto che le condizioni imposte da un programma possano essere soddisfatte da più di una funzione, il metodo di calcolo che consiste nel sostituire sistematicamente la parte sinistra di un'equazione con la parte destra definisce la *minima* di tale funzioni parziali (che è ricorsiva, in quanto calcolabile mediante un computer). Così il programma

$$\varphi(x) \simeq \varphi(x)$$

definisce la singolare funzione che non è mai definita, e la stessa cosa fa il programma

$$\varphi(x) \simeq 2 \cdot \varphi(\mathcal{S}(x))$$

considerato nella Sezione 4.2.

In precedenza, nel 1938 Kleene aveva stabilito l'analogo del secondo metodo per il trattamento delle equazioni visto sopra, mediante il cosiddetto **Teorema del Punto Fisso**. Esso dice che ogni equazione definente una funzione, in cui la variabile sia un numero per un programma della funzione stessa, ammette una soluzione numerica (che codifica una funzione ricorsiva parziale). Ad esempio, esiste una funzione ricorsiva parziale  $\varphi_e$  tale che

$$\varphi_e(x) \simeq e,$$

e che quindi ha come effetto quello di stampare un suo programma (o, meglio, un numero che codifica un suo programma).

Questa forma di autoreferenza è molto più sofisticata della precedente, e più difficile da visualizzare. Ma è interessante cercare di farlo, perchè nella soluzione è racchiuso addirittura il 'mistero della vita'.

Le idee del Teorema del Punto Fisso (in particolare, di un programma che stampi sè stesso) sono ben illustrate dal problema di costruire un meccanismo che si autoriproduca. A prima vista questo sembrerebbe impossibile: una macchina dovrebbe poter riprodurre soltanto macchine più semplici, perchè per riprodurre un'altra deve contenere in qualche modo una completa descrizione di questa, oltre ad altre parti che le permettano la riproduzione.

Su questo problema si arenò Descartes: egli voleva descrivere l'universo in maniera puramente meccanicista, eccezion fatta per l'esistenza di Dio e

dell'anima, ma non riuscì a capire come si potesse giustificare meccanicamente la riproduzione di organismi da organismi simili ad essi, e fu costretto ad invocare un miracolo per ciascuna di tali riproduzioni.

In realtà, la soluzione non è poi così difficile: *una macchina può riprodurre se stessa, quando sia capace di costruire macchine seguendone una descrizione*. Quando si raggiunga una tale complessità minimale, una macchina può non solo autoriprodursi (se le si fornisce una descrizione di sè stessa), ma anche produrre macchine più complicate. Vediamo più da vicino come questo si possa realizzare.

La prima macchina di cui abbiamo bisogno è un *costruttore universale*  $A$  che abbia la proprietà di leggere la descrizione  $d_X$  di una macchina  $X$ , di cercare nell'ambiente circostante le parti necessarie, e di costruire  $X$  seguendo la descrizione  $d_X$ . In simboli:

$$(A, d_X) \longrightarrow X.$$

La macchina  $A$  di per sè stessa non ha ancora la capacità di autoriprodursi, perchè se anche le si dà una sua descrizione  $d_A$  si ottiene sì  $A$  stessa, ma partendo non da  $A$ , bensì da  $A$  più la descrizione  $d_A$ :

$$(A, d_A) \longrightarrow A.$$

Basta però alla fine inserire in  $A$  la descrizione  $d_A$ . Per far questo abbiamo anzitutto bisogno di una macchina  $B$  che, data una descrizione  $d_X$ , ne faccia una copia:

$$(B, d_X) \longrightarrow d_X.$$

Infine, c'è bisogno di una macchina  $C$  che coordini le due macchine  $A$  e  $B$ , come segue: data una descrizione  $d_X$ ,  $B$  ne fa una copia, e  $A$  costruisce una copia di  $X$  seguendo  $d_X$ ; infine, la copia di  $d_X$  viene inserita nella copia di  $X$ . Se chiamiamo  $D$  la macchina risultante, allora

$$(D, d_X) \longrightarrow (X, d_X),$$

e se a  $D$  viene data una descrizione di sè stessa, si ha ora autoriproduzione di  $S = (D, d_D)$ :

$$(D, d_D) \longrightarrow (D, d_D).$$

Le stesse idee permettono non soltanto autoriproduzione, ma anche *processi evolutivi*. E' infatti sufficiente inserire in  $D$  la descrizione  $d_{D+F}$  di una macchina più complicata di  $D$  stessa, per avere

$$(D, d_{D+F}) \longrightarrow (D + F, d_{D+F}),$$

ed ottenere dunque una copia arricchita della macchina di partenza.

L'aspetto più interessante di questo meccanismo è che si scoprì in seguito che (come Von Neumann aveva ipotizzato nel 1951) *la vita reale si riproduce esattamente in tale modo*: le cellule viventi contengono costruttori universali,

in pratica gli stessi per piante ed animali, e solo il materiale genetico (la descrizione) è differente. La costruzione precedente si può dunque reinterpretare oggi in senso biologico, nel seguente modo:  $d_X$  (nella pratica un *gene*, cioè un segmento di *DNA*) codifica l'informazione necessaria per la riproduzione;  $B$  (nella pratica un *enzima*) ha la funzione di duplicare il materiale genetico (in un segmento di *RNA*);  $A$  (nella pratica un sintetizzatore di *proteine*) segue le informazioni contenute nell'*RNA* riprodotto;  $S$  è una *cellula* che si autoriproduce.

Il meccanismo del Teorema del Punto Fisso è dunque risultato essere lo stesso che la natura segue (senza miracoli, con buona pace di Descartes) per la riproduzione biologica.

## 5 Estensioni

La Teoria della Ricorsività, partita da problematiche strettamente filosofiche (discusse nella Sezione 3), andò muovendosi verso risultati più tecnici, benchè ancora motivati da problemi di natura generale (esposti nella Sezione 4). Dalla fine degli anni '40 quest'ultima direzione prese il sopravvento, e gli aspetti tecnici divennero predominanti. In particolare, la Teoria della Ricorsività divenne una branca della matematica a sè stante, di cui ora descriveremo alcuni aspetti.

### 5.1 Oracoli

La classificazione delle funzioni (totali) in ricorsive e non ricorsive è fondamentale, ma piuttosto grossolana: infatti, *la maggioranza delle funzioni consiste di funzioni non ricorsive*.

Si può dire che *l'infinito delle funzioni ricorsive è dello stesso ordine di grandezza dell'infinito dei numeri*, perchè i numeri si possono porre in una lista

$$0 \quad 1 \quad 2 \quad \dots,$$

e così si può fare per le funzioni ricorsive totali, perchè esse sono una sottolista della lista

$$\varphi_0 \quad \varphi_1 \quad \varphi_2 \quad \dots$$

delle funzioni ricorsive parziali, ottenuta eliminando da essa le funzioni che non sono totali (il fatto che non ci sia nessun metodo ricorsivo per fare ciò, come abbiamo visto nella sezione 4.3, significa soltanto che la lista finale non sarà ricorsiva).

D'altra parte, *l'infinito delle funzioni è di un ordine più grande dell'infinito dei numeri*, perchè una qualunque lista di funzioni totali

$$f_0 \quad f_1 \quad f_2 \quad \dots$$

non le può contenere tutte. Ad esempio, come ormai abbiamo visto più volte, non può contenere la funzione

$$f(x) = f_x(x) + 1$$

definita per diagonalizzazione (tale metodo fu appunto introdotto da Cantor per provare nel 1874 questo risultato, che segnò la nascita dello studio moderno dell'infinito).

Nel 1939 Turing introdusse un metodo che permette di discriminare fra loro funzioni non ricorsive (e dunque problemi insolubili), e di distinguere vari **gradi di insolubilità**. L'idea consiste nel passare dalla nozione di calcolabilità assoluta, usata finora, a quella di calcolabilità relativa ad un certo **oracolo**, a cui si possa chiedere aiuto durante un calcolo. Tale idea si può pensare come un'astrazione del *processo interattivo uomo-macchina*, in cui il computer ha la possibilità di demandare certe scelte al programmatore: tale possibilità è utile quando, per un qualunque motivo, non si voglia o non si sappia affidare ogni parte di un calcolo al computer (ad esempio, come il film *War Games* insegna, quando l'effetto del calcolo è un attacco nucleare può essere consigliabile lasciare al programmatore la valutazione di alcuni fattori).

Due problemi si dicono dello *stesso grado* se ciascuno si può risolvere usando l'altro come oracolo (ed essi hanno quindi la stessa difficoltà); un problema si dice invece di *grado inferiore* ad un altro se il primo è risolubile usando il secondo come oracolo, ma non viceversa (e dunque il primo è più facile da risolvere del secondo). Ad esempio, i due seguenti problemi hanno lo stesso grado di insolubilità:

- decidere, dato un programma, se esso calcola la funzione costante di valore 0
- decidere, dati due programmi, se essi calcolano la stessa funzione.

Il primo problema è risolubile relativamente al secondo: se sappiamo decidere quando due programmi calcolano la stessa funzione, e vogliamo decidere se un dato programma calcola la funzione costante 0, basta confrontarlo con un qualunque programma che calcoli tale funzione (ad esempio:  $f(x) = 0$ ).

Viceversa, il secondo problema è risolubile relativamente al primo: se sappiamo decidere quando un programma calcola la funzione costante di valore 0, e vogliamo decidere se due programmi calcolano la stessa funzione, basta vedere se il programma che calcola la differenza delle due funzioni calcola la funzione costante di valore 0.

Una buona parte dei risultati della Teoria della Ricorsività consiste nello studio, iniziato da Kleene e Post nel 1954, delle proprietà di struttura dei gradi. Ad esempio, è ovvio che esiste un *grado minimo* (quello dei problemi che sono risolubili, senza chiedere aiuto a nessun oracolo; equivalentemente, quello delle funzioni ricorsive).

Meno ovvio è che ci non ci siano problemi di *grado massimo*. Questo risulta per analogia col (in termine tecnico, per **relativizzazione** del) fatto che il Problema della Fermata non è risolubile: dato un qualunque problema, si può considerarlo come un oracolo, ed il Problema della Fermata per programmi che

usino tale oracolo non è risolubile relativamente ad esso (ed ha, in realtà, grado maggiore).

Ancor meno ovvio è che ci siano problemi con *gradi inconfrontabili*, e quindi di difficoltà incomparabile. O problemi con *gradi minimali*, cioè non risolubili, e tali che non possono fungere in modo utile da oracoli per nessun altro problema (gli unici problemi risolubili con tali oracoli sono quelli della stessa difficoltà, e quelli risolubili senza oracoli). Le dimostrazioni usano il metodo di diagonalizzazione, con una buona dose di inventiva.

Come i risultati citati incominciano a far sospettare, si è scoperto che *la struttura dei gradi è estremamente complicata*: questa in un certo senso è stata una delusione, perchè si sperava che lo studio dei gradi avrebbe potuto portare ad una descrizione semplice dell'insieme di tutte le funzioni.

Nonostante oggi si conosca molto sulla struttura dei gradi, una sua completa caratterizzazione non è ancora stata trovata. L'obiettivo è, come spesso in matematica, di riuscire ad isolare proprietà della struttura che la determinino in modo univoco (sulla falsariga di quanto fatto da Dedekind, e ricordato nella Sezione 1.2, per i numeri naturali).

## 5.2 Il Problema di Post

Finora abbiamo parlato di funzioni e di loro classificazioni. Ma una gran parte della Teoria della Ricorsività consiste nello studio e nella classificazione degli **insiemi** di numeri, che possono essere visti come particolari funzioni a valori 0 ed 1: tali valori possono essere interpretati come risposte positive o negative alla domanda se l'argomento stia o no nell'insieme dato. In particolare, anche per gli insiemi si può parlare di calcolabilità relativa, e di loro gradi di insolubilità.

Gli insiemi il cui grado è il più semplice possibile sono quelli detti **ricorsivi**, tali cioè che la funzione ad essi associata sia ricorsiva. Essi sono dunque quelli per i quali un computer può determinare se un elemento sta oppure no nell'insieme.

Un tipo particolarmente interessante di insiemi, non necessariamente ricorsivi, ma ancora con caratteristiche di effettività, sono gli insiemi **ricorsivamente enumerabili**: come dice la parola, essi sono gli insiemi per i quali esiste una funzione ricorsiva che ne enumera (in qualunque ordine ed, eventualmente, con ripetizioni) gli elementi. Essi sono dunque quelli i cui elementi possono essere enumerati da un computer.

Se un insieme è ricorsivo, allora sappiamo completamente rispondere a domande riguardanti l'appartenenza di un elemento ad esso. Se un insieme è ricorsivamente enumerabile, abbiamo soltanto un metodo che ci permette di rispondere affermativamente nel caso che la risposta sia positiva: generare l'insieme, fino a quando l'elemento in questione appaia nella lista. Ma tale metodo non permette di accorgersi se un dato elemento non sta nell'insieme: durante il processo di generazione, il fatto che un elemento non sia ancora apparso nella lista non permette di concludere che esso non comparirà in seguito.

Le osservazioni precedenti mostrano come però ci sia un collegamento fra le due nozioni. Ad esempio, *un insieme infinito che venga generato in ordine crescente è ricorsivo*: per sapere se  $x$  sta nell'insieme oppure no, basta generare l'insieme fino a che un elemento maggiore di  $x$  appare, e poi vedere se  $x$  è nella lista parziale ottenuta finora.

Analogamente, *se non solo un insieme ma anche il suo complemento* (cioè l'insieme degli elementi che non stanno in esso) *è ricorsivamente enumerabile, allora l'insieme è ricorsivo*: per sapere se  $x$  sta nell'insieme oppure no, basta generare simultaneamente l'insieme e il suo complemento, fino a che  $x$  appaia in una delle due liste.

L'insolubilità del Problema della Fermata prova che *esistono insiemi ricorsivamente enumerabili ma non ricorsivi*.

Il Teorema di Enumerazione fornisce un programma che, dati  $x$  e  $z$ , simula il programma  $x$  sull'argomento  $z$ . Per generare una lista di tali coppie  $(x, z)$ , e dunque mostrare che il loro insieme è ricorsivamente enumerabile, si usano due accorgimenti. Anzitutto, si enumerano tutte le possibili coppie  $(x, z)$ , ad esempio in base alla loro somma  $x + y$ , e in base alla prima componente se la somma è la stessa:

$$\underbrace{(0, 0)}_{\text{somma 0}} \quad \underbrace{(0, 1) \quad (1, 0)}_{\text{somma 1}} \quad \underbrace{(0, 2) \quad (1, 1) \quad (2, 0)}_{\text{somma 2}} \quad \dots$$

Si tratta ora di generare una lista delle coppie  $(x, z)$  per cui il programma  $x$  si ferma sull'argomento  $z$ , e il problema è che non si può semplicemente considerare una coppia per volta e simulare  $x$  su  $z$ , perchè non appena si incontra un programma che non si ferma, questo impedirebbe di considerare le rimanenti coppie.

L'idea è di non essere troppo precipitosi, e di procedere per approssimazioni: consideriamo la prima coppia per un minuto, poi le prime due coppie per due minuti ciascuna, poi le prime tre coppie per tre minuti ciascuna, e così via. Se e quando ci accorgiamo che il programma  $x$  si ferma sull'argomento  $z$  (e se così è ce ne accorgeremo prima o poi, perchè ciascuna coppia viene considerata per tempi via via più lunghi), enumeriamo la coppia  $(x, z)$  nella lista. Questo prova che il Problema della Fermata è ricorsivamente enumerabile.

Poichè per ogni insieme ricorsivamente enumerabile esiste un programma  $x$  che si ferma sull'argomento  $z$  se e solo se esso sta nell'insieme (basta definire  $x$  in modo tale che dia come risultato 0 se  $z$  viene generato nell'insieme, e non si fermi altrimenti), *il Problema della Fermata ha il massimo grado fra quelli degli insiemi ricorsivamente enumerabili*.

Nel 1944 Post chiese se ci fossero soltanto due gradi di insiemi ricorsivamente enumerabili, cioè quello dei problemi risolubili, e quello del Problema della Fermata. Tale domanda divenne nota come **Problema di Post**, e fu risolta negativamente nel 1956 da Richard Friedberg (n. 1935) e Muchnik, che inventarono una modifica costruttiva del metodo di diagonalizzazione, detta **metodo di priorità**. Essi provarono, in realtà, che anche in questo caso esistono insiemi di

grado inconfondibile.

La teoria dei gradi di insiemi ricorsivamente enumerabili è la parte più sofisticata e difficile della Teoria della Ricorsività, e la struttura di tali gradi è ancora più complessa e patologica di quella dei gradi in generale. Dunque, ancora più di questa, essa è risultata una delusione.

### 5.3 Analisi ricorsiva

Finora ci siamo limitati a studiare funzioni e insiemi di numeri. Come però alcuni degli esempi considerati hanno mostrato, la nozione di algoritmo non è limitata a procedimenti strettamente numerici. Nella pratica, un algoritmo può riferirsi ad oggetti della più svariata natura o, in linguaggio informatico corrente, a *dati di varia struttura*.

A partire dalla fine degli anni '50, e con un interesse via via più crescente, la Teoria della Ricorsività ha cercato di svincolarsi dall'approccio puramente numerico, e di estendere la propria ricerca alla nozione di computabilità su oggetti di varia natura.

Una tale estensione è ovviamente richiesta quando si vogliono considerare operazioni sui numeri reali, come nell'analisi. Per poter parlare di calcolabilità di tali operazioni si deve estendere la nozione di calcolabilità da funzioni di numeri naturali a funzioni di numeri reali, e questo ha portato allo sviluppo dell'**analisi ricorsiva**. In realtà, ci sono due diversi approcci a tale argomento.

Anzitutto, si può restringersi a considerare soltanto numeri reali il cui sviluppo decimale (visto come una funzione numerica che assegna all'argomento  $n$  la  $n$ -sima cifra dello sviluppo decimale del numero reale considerato) sia ricorsivo: tali numeri si chiamano **reali ricorsivi**, e si è scoperto che essi costituiscono una ricca parte dell'insieme dei numeri reali. In particolare, tutti i numeri reali che sono usati in pratica (ad esempio  $\sqrt{2}$ ,  $\pi$ ,  $e$ ) sono risultati essere ricorsivi. Poichè ad ogni numero reale ricorsivo si può associare un programma che ne stampa lo sviluppo decimale, e a tale programma si può associare un numero per aritmetizzazione, si può associare ad ogni funzione di reali ricorsivi una funzione di numeri (degli associati programmi), e si può quindi parlare di **funzioni ricorsive di reali ricorsivi** quando le funzioni di numeri ad esse associate lo sono.

Un altro approccio all'analisi ricorsiva consiste nel definire una nozione di **funzione ricorsiva di reali qualunque**. Poichè, come abbiamo notato, i numeri reali si possono vedere come funzioni di numeri naturali, tale approccio è strettamente legato a quella che si chiama Teoria della Ricorsività per **oggetti di tipo superiore**, dove per oggetti di tipo 0 si intendono i numeri, per oggetti di tipo 1 le funzioni su numeri (che si possono identificare, pensandone i valori come cifre successive di uno sviluppo decimale, con numeri reali), e così via.

Una tale teoria presenta interessanti aspetti, in particolare il fatto che non ci si può più aspettare, come nel caso della calcolabilità su numeri (o anche su numeri reali ricorsivi), che un calcolo richieda soltanto un tempo finito: in fin

dei conti, gli oggetti su cui si lavora (ad esempio, i numeri reali) sono già essi stessi di natura infinita. E' stato dunque necessario analizzare la nozione stessa di **finitezza**, e capire quali delle sue proprietà sono essenziali per una nozione astratta di calcolo, e quali invece si possono lasciar cadere senza detrimento.

Un tipico risultato dell'analisi ricorsiva, che mostra quanto essa possa scostarsi dall'analisi classica, è che *ogni funzione ricorsiva di numeri reali è continua*.

## 6 Ritorno alle Origini

Parallelamente alle estensioni tecniche e alle astrazioni menzionate nella Sezione 5, lo sviluppo della Teoria della Ricorsività non ha dimenticato le sue origini legate ai computers e alla nozione di calcolabilità. In particolare, a partire dagli anni '60 l'analisi della nozione di computazione è stata ulteriormente approfondita, ed ha dato origine ad una nuova branca detta **Teoria della Complessità**, il cui obbiettivo è quello di analizzare la calcolabilità non più solo da un punto di vista astratto, come nel caso della ricorsività, ma anche da un punto di vista concreto o pratico.

### 6.1 Casualità

Un primo approccio consiste nel notare che l'universo è finito sia spazialmente che temporalmente (se non in senso assoluto, almeno relativamente alle possibilità di fruizione dell'uomo stesso), e che dunque in pratica si possono calcolare veramente soltanto **funzioni finite**. Uno degli obiettivi della Teoria della Complessità è stato quindi quello di classificare la complessità di oggetti finiti e dunque (mediante una loro aritmetizzazione) di numeri.

Una possibile misura della complessità di un numero si ottiene considerando la sua più corta descrizione (o, più astrattamente, il più piccolo numero che codifica un programma senza argomenti che stampa il numero stesso). Si arriva così alla nozione di **numero casuale**, definito come un numero  $x$  tale che ogni sua descrizione abbia lunghezza (o, più astrattamente, ogni programma che lo stampi sia codificato da un numero)  $\geq x$ .

E' facile esibire numeri non casuali: ad esempio, il 'numero che consiste di un 1 seguito da nove 0' è stato or ora descritto usando 10 parole (o 45 caratteri, inclusi gli spazi fra parole), ma il suo valore numerico è un miliardo.

Meno facile è esibire numeri casuali, proprio perchè il modo più spiccio di descriverli è quello di scrivere le loro cifre, la cui successione deve essere sufficientemente caotica, in modo da non permettere una descrizione compatta. Ma si può facilmente convincersi che *ci sono infiniti numeri casuali*.

Dato un numero  $n$ , si considerino i programmi senza argomenti ordinati in base ai numeri che li codificano, e si considerino i numeri stampati dai primi  $n + 1$  di tali programmi. Se  $x$  è diverso da tali numeri, la sua complessità deve essere almeno  $n + 1$ .

Se inoltre  $x$  è il minimo di tale numeri, esso è  $\leq n + 1$ , perchè abbiamo considerato  $n + 1$  programmi, e nel caso peggiore essi stampano esattamente tutti i numeri fino ad  $n$  (non  $n + 1$ , perchè c'è anche lo 0 da considerare). Allora tale  $x$  è  $\leq n + 1$ , ma la sua complessità è  $\geq n + 1$ : dunque  $x$  è casuale. Poichè c'è un numero casuale di complessità almeno  $n + 1$ , ed  $n$  è qualunque, ci sono infiniti numeri casuali.

Una conseguenza interessante della nozione di casualità è che essa mostra come sia impossibile *distinguere tra caos e pianificazione esagerata*: un numero può essere casuale perchè le sue cifre si ottengono tirando un dado, o perchè le condizioni su di esse sono così complicate da non poter essere descritte brevemente.

Simmetricamente, l'esistenza di oggetti casuali mette in guardia dal credere di poter sempre ottenere una *descrizione efficiente di qualunque oggetto*. Infatti, la più semplice descrizione di un oggetto casuale consiste nell'esibire l'oggetto stesso direttamente. Ad esempio, Von Neumann indicò il cervello come un possibile oggetto casuale, per la descrizione del quale potrebbero dunque esistere insormontabili difficoltà intrinseche.

La nozione di numero casuale permette di indicare un'altra limitazione dei programmi, analoga a quelle già viste nella Sezione 4.3: *nessun programma che generi soltanto numeri casuali ne può generare un numero infinito*.

Sia dato un programma che stampa infiniti numeri, e proviamo che non tutti possono essere casuali. Esiste un numero  $e$  tale che il programma di numero  $e$  stampa il più piccolo numero maggiore di  $e$  generato dal programma dato (per il Teorema del Punto Fisso). Ma allora tale numero è stampato da un programma di numero minore di esso, dunque non è casuale, ed è anche generato dal programma dato. Questo dunque non stampa soltanto numeri casuali.

Come già nella Sezione 4.4, la limitazione appena discussa può essere riformulata in una versione del Teorema di Gödel: *un sistema di assiomi e regole che sia codificabile mediante un programma e che non menta mai può provare la casualità soltanto di un numero finito di numeri casuali* (altrimenti si potrebbero generare infiniti numeri casuali generando i teoremi del sistema, e considerando i numeri di cui il sistema prova la casualità).

## 6.2 Calcolabilità pratica

Anche se nella pratica le sole funzioni che sono veramente calcolabili sono quelle finite, le funzioni solite sono utili per considerare comportamenti al limite (in termini tecnici, *asintotici*). Infatti, nella pratica le funzioni (e non solo di numeri naturali, ma addirittura di numeri reali) sono usate correntemente nello studio di fenomeni finiti, ad esempio in fisica o in economia.

Un approccio alternativo a quello della Sezione 6.1 consiste dunque nel mantenere la nozione di funzione ricorsiva, come descrizione della nozione di calco-

labilità astratta, e di cercare restrizioni che isolino la nozione di calcolabilità pratica.

Un possibile approccio è quello di classificare le funzioni ricorsive in base a varie **misure di complessità**, che ne quantifichino la difficoltà relativamente ad alcuni parametri considerati significativi. Sono state proposte varie misure, ciascuna basata su un particolare aspetto della nozione di calcolo pratico: di richiedere un certo *tempo*, o una certa quantità di *memoria*, o un certo numero di *esecuzioni di istruzioni*. Ovviamente, una stessa funzione può risultare facile da calcolare rispetto ad uno di questi criteri, ma difficile rispetto ad un altro. Inoltre, lo stesso può succedere per una stessa funzione rispetto addirittura allo stesso criterio (ad esempio, il tempo richiesto), ma applicato a due diversi modelli di calcolo.

L'esperienza con la nozione di calcolabilità ci porta però a non disperare, ed infatti anche i fondamenti della Teoria della Complessità hanno riservato un'altra sorpresa: *un gran numero di misure di complessità rispetto ad un gran numero di modelli di calcolo (compresa, fra esse, la misura di tempo su un computer) differiscono soltanto per un fattore polinomiale*. La classe delle **funzioni calcolabili in tempo polinomiale** (nella lunghezza degli argomenti) possiede dunque una buona invarianza, ed è oggi considerata come comprendente tutte (anche se, certamente, non sole) le funzioni calcolabili praticamente. Ad esempio, i soliti algoritmi per la somma e il prodotto mostrano come la somma sia calcolabile in un numero di operazioni che è proporzionale alla lunghezza massima dei due addendi, e il prodotto in un numero che è proporzionale al quadrato di questa; esse sono dunque calcolabili in tempo polinomiale. D'altra parte, l'esponenziazione richiede invece molto più tempo, e si può dimostrare che non è calcolabile in tempo polinomiale.

### 6.3 Determinismo

Data un'espressione numerica, esistono semplici regole che ne determinano in modo univoco il valore, senza però specificare l'ordine di operazioni da seguire. Ad esempio, di fronte ad un'espressione del tipo  $0 \cdot 2^x$  si può decidere di calcolare prima il valore di  $2^x$  (il che, come abbiamo appena notato, non si può fare in tempo polinomiale nella lunghezza di  $x$ ) e poi di moltiplicarlo per 0, o si può usare immediatamente il fatto che un qualunque numero moltiplicato per 0 dà 0 (ottenendo così il risultato in un passo). Quando tali regole vengono tradotte per i calcolatori, esse danno luogo a **programmi non deterministici**.

Le regole del calcolo numerico sono particolarmente semplici, nel senso che qualunque ordine di esecuzione porta ad uno ed un solo risultato; la scelta della strategia da seguire è dunque, in questo caso, soltanto un problema di efficienza. Ma esistono situazioni più complicate, in cui soltanto qualche ordine di esecuzione porta ad un risultato. Ad esempio, se si considera un programma che codifica un sistema di assiomi e regole, e si è interessati a sapere se una certa formula è un teorema, non serve generare dimostrazioni a caso. Per ot-

tenere il risultato voluto, nel caso in questione una dimostrazione della formula data, si deve seguire un particolare ordine di esecuzione, altrimenti si rischia di non raggiungere mai il risultato, anche quando questo esiste. Naturalmente, se il risultato non esiste anche il generare sistematicamente tutte le possibili dimostrazioni non porterà ad una risposta: questa situazione è tipica degli insiemi ricorsivamente enumerabili, discussi nella Sezione 5.2.

Se, come naturale dal punto di vista della Teoria della Complessità, si considerano sì programmi non deterministici, ma li si lasciano operare soltanto per un tempo polinomiale, si ottiene una situazione interessante: se il risultato esiste, esso si può trovare (nel caso che si facciano le scelte giuste durante l'esecuzione) in tempo polinomiale; ma se il risultato non esiste, questo si può scoprire soltanto provando ogni possibile scelta, e dunque in tempo molto più lungo (in generale, non polinomiale). La dicotomia fra 'finito' e 'infinito' (o possibile e impossibile da un punto di vista *astratto*) che era tipica degli insiemi ricorsivamente enumerabili, viene così sostituita dalla dicotomia fra 'polinomiale' e 'non polinomiale' (o possibile e impossibile da un punto di vista *pratico*).

Un successo della Teoria della Complessità nel 1971 è stato quello di provare che, così come il Problema della Fermata ha la massima difficoltà fra quelli ricorsivamente enumerabili, così *esistono problemi della massima difficoltà fra quelli risolvibili in tempo polinomiale con programmi non deterministici*. Un esempio è il **Problema del Commesso Viaggiatore**: trovare, dato un numero finito di città collegate da strade, un percorso che passi da ciascuna città una ed una sola volta.

Ci si può chiedere se, così come il Problema della Fermata non è risolvibile ricorsivamente, così il Problema del Commesso Viaggiatore non sia risolvibile in tempo polinomiale deterministico. La risposta a tale problema, chiamato  $P = NP$  (dove  $P$  sta per 'polinomiale', e  $NP$  per 'polinomiale non deterministico'), è però ancora sconosciuta.

Il problema è diventato uno fra i più importanti dell'informatica teorica, e non soltanto per motivi matematici: come l'esempio del Problema del Commesso Viaggiatore lascia intravedere, molti problemi di interesse pratico si sanno risolvere in tempo polinomiale non deterministico, ma non in tempo polinomiale deterministico. In altre parole, se ne conoscono soluzioni 'quasi' efficienti ma non soluzioni efficienti: poichè essi vengono usati nella vita quotidiana, tale inefficienza si traduce in un costo elevato. Una soluzione positiva al problema  $P = NP$  potrebbe dunque avere grandi benefici economici.

## 7 Conclusione

La Teoria della Ricorsività si situa in un'area di contatto fra filosofia, matematica e informatica. Dalla *filosofia della scienza* ha tratto l'ispirazione iniziale, e ad essa ha contribuito mediante un'analisi epistemologica di concetti, primo fra tutti quello di 'calcolabilità'. Dalla *matematica* ha attinto metodologie e

problematiche, e ne è diventata parte integrante come studio delle funzioni di numeri naturali. Dell'*informatica* è stata la progenitrice, e continua ad essere il riferimento teorico per questioni connesse alla calcolabilità. In tal senso essa si situa all'interno della *logica matematica* non soltanto come una delle sue quattro parti fondamentali (insieme a Teoria degli Insiemi, Teoria dei Modelli, e Teoria della Dimostrazione, che sono l'oggetto dei Capitoli 6, 7, 11 e 12 di questo volume), ma anche come un suo solido collegamento con il fluire dello sviluppo scientifico moderno.

## Bibliografia

Trattamenti dei fondamenti della Teoria della Ricorsività si possono trovare in un gran numero di testi. Ad esempio, i manuali di Logica Matematica dedicano in genere ad essi uno o più capitoli. Tra le opere tradotte in italiano, rinviamo a:

- Hermes, H., *Enumerabilità, decidibilità, computabilità*, Boringhieri, 1977.
- Börger, E., *Computabilità, complessità, logica*, Boringhieri, 1990.

Benchè non propriamente un testo di logica in generale o di ricorsività in particolare, il seguente è un libro stimolante e non tecnico che tratta di alcuni degli argomenti toccati in questo capitolo:

- Hofstadter, D., *Gödel, Escher e Bach*, Adelphi, 1984.

Per le parti più avanzate della Teoria della Ricorsività, due testi (di cui non esistono traduzioni in italiano) coprono l'intero spettro degli argomenti trattati in questo capitolo:

- Rogers, H., *Theory of recursive functions and effective computability*, McGraw Hill, 1967.
- Odifreddi, P., *Classical Recursion Theory*, North Holland, 1989 (volume I) e 1996 (volume II).

Nelle opere citate si possono poi trovare un gran numero di suggerimenti per ulteriori letture, così come i riferimenti alle pubblicazioni originali in cui sono apparsi i risultati da noi citati.