DENTRO LA SCATOLA

Rubrica a cura di

Fabio A. Schreiber

Il Consiglio Scientifico della rivista ha pensato di attuare un'iniziativa culturalmente utile presentando in ogni numero di Mondo Digitale un argomento fondante per l'Informatica e le sue applicazioni; in tal modo, anche il lettore curioso, ma frettoloso, potrà rendersi conto di che cosa sta "**dentro la scatola**". È infatti diffusa la sensazione che lo sviluppo formidabile assunto dal settore e di conseguenza il grande numero di persone di diverse estrazioni culturali che - a vario titolo - si occupano dei calcolatori elettronici e del loro mondo, abbiano nascosto dietro una cortina di nebbia i concetti basilari che lo hanno reso possibile.
Il tema scelto per il 2004 è stato: "***Perché gli anglofoni lo chiamano computer***, ovvero: ***introduzione alle aritmetiche digitali***". Per il 2005 il filo conduttore della serie sarà: "***Ma ce la farà veramente?***, ovvero: ***introduzione alla complessità computazionale e alla indecidibilità***" e il suo intento è di guidare il lettore attraverso gli argomenti fondanti dell'Informatica e alle loro implicazioni pratiche e filosofiche. La realizzazione degli articoli è affidata ad autori che uniscono una grande autorevolezza scientifica e professionale a una notevole capacità divulgativa.

Potenza e limiti del calcolo automatico: le radici teoriche dell'informatica

Dino Mandrioli

INTRODUZIONE

Questo primo articolo della serie dedicata agli aspetti più concettuali e profondi dell'informatica ha un taglio decisamente storico: mostreremo come, contrariamente al creder comune, l'informatica non sia affatto "nata ieri" e non consista soltanto in un agglomerato inestricabile di circuiti elettronici e software; che non offra solo "servizi web", giochi elettronici, impianti satellitari ecc., ma affondi le sue radici nelle stesse origini del pensiero umano e abbia contatti e intersezioni fortissimi con discipline "nobili e astratte" come la matematica e la filosofia.
Dopo un breve richiamo al concetto di algoritmo e alla sua essenza indissolubilmente legata all'esecuzione mediante uno strumento automatico ne investigheremo non solo la potenza ma anche i limiti in termini dei problemi che attraverso questo strumento possono essere affrontati e risolti. Sottolineeremo come potenza e limiti del calcolo automatico abbiano un fondamentale impatto sia in termini pratici sulla nostra attività quotidiana sia sulla natura stessa del pensiero umano.
Il successivo articolo di questa serie, pur mantenendo uno stile divulgativo, entrerà nel merito delle tecniche utilizzate per arrivare ai risultati fondamentali qui enunciati: ciò darà ulteriore evidenza alle connessioni tra informatica, matematica e filosofia.

ALGORITMI; STRUMENTI E MODELLI DI CALCOLO AUTOMATICO

Tra le tante formulazioni del concetto di algoritmo facciamo riferimento alla seguente:

> *"Un algoritmo è un processo di elaborazione di informazioni, basato su una codifica rigorosa e precisa delle medesime e costituito da una sequenza di passi elementari definita in maniera altrettanto precisa e rigorosa, tanto da non lasciare alcun dubbio all'ente preposto alla sua esecuzione."*

Esempi universalmente noti e applicati di algoritmi sono gli algoritmi per il calcolo delle operazioni aritmetiche, per la ricerca di un elemento all'interno di un insieme, per mettere in ordine una sequenza di elementi ecc..
A sottolineare il fatto che questo fondamentale pilastro dell'informatica ha radici storiche millenarie, ricordiamo il famoso algoritmo di Euclide per il calcolo del massimo comun divisore (MCD) tra due numeri naturali x e y. Esso è basato sulla

constatazione che se $x = y$, allora evidentemente MCD $(x, y) = x = y$; altrimenti esso coincide con il MCD $(x - y, y)$, se $x > y$ o il MCD $(y - x, x)$ se $y > x$. Di conseguenza, continuando a sottrarre il minore tra i due all'altro e sostituendo il risultato della differenza al maggiore, si ottiene una sequenza che termina con il MCD (che sarà 1 nel caso degenere di due numeri primi tra loro). Per esempio, partendo da 15 e 9 si ottiene la sequenza <6, 9>, <3, 6>, <3, 3> e quindi MCD (15, 9) = 3.
Questa formulazione ed esemplificazione del concetto di algoritmo ci permette una prima serie di commenti e delucidazioni:

- Ogni "problema informatico", anche il più complesso come la gestione di varie forme di commercio elettronico o il monitoraggio e controllo di una rete elettrica nazionale, consiste nella costruzione di nuova informazione a partire dall'informazione esistente. Di conseguenza, ogni "progetto informatico" altro non è, in ultima analisi, che un complesso algoritmo, il quale al suo interno racchiuderà altri algoritmi per la soluzione di sottoproblemi (anche un algoritmo per il calcolo della moltiplicazione può sfruttare un sottoalgoritmo per il calcolo della somma), e così via fino alla scomposizione in un enorme numero di operazioni elementari che, nella tecnologia elettronica digitale, consistono nell'elaborazione di singoli bit.
- L'esecuzione di un algoritmo non richiede "intelligenza" ma soltanto precisione: quando sbagliamo la soluzione di un problema applicando un algoritmo, diciamo che "abbiamo sbagliato i conti" e, giustamente, tendiamo a valutare l'errore commesso come "non grave", rispetto ad errori come la cattiva progettazione dell'algoritmo o addirittura la cattiva comprensione stessa del problema.
- Proprio la precedente osservazione ha generato, fin da subito, il desiderio di usare una macchina per eseguire algoritmi: sono ben noti, infatti, i vari "prototipi di calcolatori" costruiti nei secoli da diversi "pionieri" come Blaise Pascal, Charles Babbage e vari altri.
- È altresì noto che per molti secoli il calcolo meccanico sia rimasto un sogno e divenne realtà pratica solo relativamente di recente con l'avvento della tecnologia elettronica digitale.

Traendo spunto da quest'ultima osservazione, è importante notare che "la fantasia umana ha largamente anticipato l'evoluzione tecnologica"; in altri termini *modelli di calcolatori*, ossia *macchine astratte* per l'esecuzione di algoritmi, sono stati concepiti e studiati a fondo ben prima che i moderni calcolatori elettronici facessero il loro ingresso nel mondo industriale. Tra i tanti, la *Macchina di Turing*, che deve il nome al suo inventore, è forse il modello più semplice e affascinante di calcolatore. Come suggerito dalla figura 1, essa consiste in due parti:

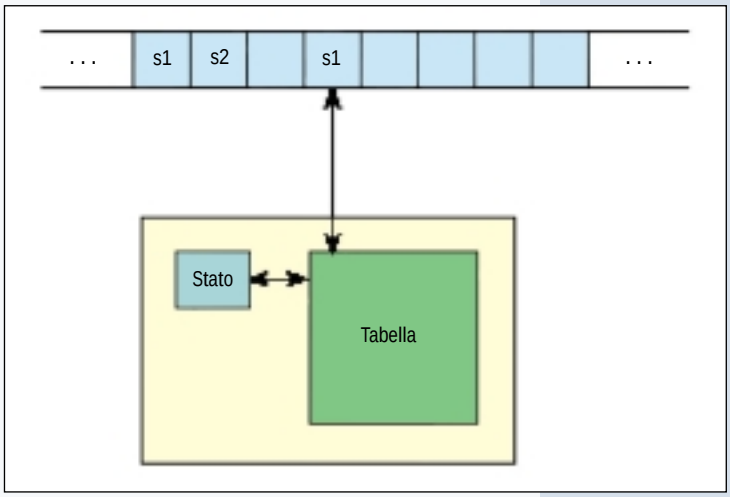

FIGURA 1
Una macchina di Turing

- un *nastro* infinito, suddiviso in celle, ciascuna contenente un *singolo simbolo* (come una lettera dell'alfabeto o una cifra decimale). Il nastro funge da *supporto per l'inserimento dei dati* (contiene i dati da elaborare), da *supporto di memoria* (è possibile leggere e cambiare il contenuto delle celle), e da *supporto di uscita* (il risultato del calcolo è parte di quello che rimane sul nastro quando l'esecuzione ha termine);
- un'unità di controllo, che in ogni istante si trova in uno e un solo *stato*, appartenente ad un prefissato insieme finito.

Un'operazione elementare di una macchina di Turing consiste nella:

- lettura di un dato dal nastro per mezzo di una *testina di lettura-scrittura* che mette in comunicazione l'organo di controllo con il nastro;
- lettura dello stato dell'organo di controllo.

In base alle letture effettuate la macchina:

- scrive un nuovo simbolo sul nastro al posto di quello letto;
- modifica lo stato dell'organo di controllo;
- sposta la testina di lettura-scrittura di una posizione a destra o a sinistra, o la lascia dove si trova.

In alternativa, la macchina viene posta in stato di *halt*, in cui cioè non è in grado di eseguire altri movimenti; a questo punto l'elaborazione è terminata.

L'algoritmo da eseguire viene comunicato alla macchina di Turing attraverso una tabella rettangolare in cui le righe indicano i simboli che possono essere letti dal nastro, le colonne indicano i possibili stati, e ogni cella indica l'operazione da eseguire in funzione del simbolo letto e dello stato dell'organo di controllo. Una cella contiene perciò una terna di elementi <simbolo da scrivere sul nastro, nuovo stato, spostamento della testina>; una cella vuota indica che la macchina si deve arrestare.

La figura 2 descrive, a titolo di esempio, una semplice macchina il cui alfabeto del nastro consiste in due soli simboli: "|" e "–", e il cui insieme di stati è dato da s1 e s2. Nell'ipotetico stato iniziale s1, il nastro contiene un numero *n* di "|" consecutivi, tutte le altre celle contengono "–", e la testina è posta sulla prima barra a sinistra. È facile constatare che, alla fine dell'elaborazione, ci sono *n* + 1 barre sul nastro; in altri termini, la macchina calcola *n* + 1 a partire da n sulla base di una codifica unaria.

Altri modelli matematici formalizzano la costruzione del linguaggio (le *Grammatiche formali* elaborate dai linguisti-matematici sono una "versione matematica" del concetto normalmente associato a questo termine), o addirittura la formulazione stessa del pensiero: in ultima analisi la sorgente primaria di informazione è proprio il nostro pensiero. Non a caso l'origine storica della *logica matematica* risiede nella logica aristotelica.

Nella prossima sezione scopriremo gli affascinanti legami tra i disparati modelli astratti di elaborazione dell'informazione e le macchine reali che, sparse per il mondo, realizzano tale elaborazione.

Stato \ s	–	|
s1	<|, s2, R>	<|, s1, R>
s2		

FIGURA 2

Tabella di una macchina di Turing per il calcolo di n + 1 in codifica unaria

La tesi di Church e la (in)calcolabilità

Una volta elaborato il concetto di algoritmo, sviluppati i primi algoritmi per la soluzione di vari problemi, concepiti e realizzati i primi strumenti per eseguire algoritmi, sorge spontanea e impellente la domanda: "*Quanti e quali problemi è possibile risolvere mediante il calcolo automatico (ossia mediante l'esecuzione di algoritmi)*?"

A prima vista, una lettura pignola ma superficiale di questa domanda potrebbe portare a ritenerla addirittura mal posta: la definizione di algoritmo è tutt'altro che matematicamente precisa e fa riferimento a quelli che possiamo definire "passi elementari". Chiaramente, tali passi, possono variare molto a seconda che l'esecutore dell'algoritmo sia un essere umano, o un calcolatore tra i tantissimi che sono stati costruiti o saranno costruiti, o un modello astratto come la macchina di Turing. Inoltre la nozione stessa di problema può essere codificata in diverse maniere: per esempio un numero può essere rappresentato in forma unaria (mediante aste o bastoncini come abbiamo imparato alle elementari e ricordato nella sezione precedente) o in base 10 o in base 2 ecc.; chi ci garantisce che un problema che non sappiamo risolvere se formulato in una certa maniera, non diventi invece risolvibile cambiandone formalizzazione? Una semplice analisi ci aiuterà a sgombrare il campo da questi dubbi, pur giustificati.

In primo luogo osserviamo che, anche se è vero che uno stesso problema può essere formulato in diverse maniere, è sempre possibile tradurre una rappresentazione in un'altra: questa stessa traduzione è un problema di elaborazione dell'informazione e può a sua volta essere eseguita mediante opportuni algoritmi. Per esempio, sono ben noti gli algoritmi per trasformare la rappresentazione di un numero da una base ad un'altra. Quindi, se riusciamo a risolvere un problema mediante una certa formalizzazione, possiamo risolverlo anche in una formalizzazione diversa mediante un'opportuna conversione.

Ancor più affascinante, e di enorme impatto concettuale e pratico, è la constatazione che i numerosissimi modelli di calcolo nonché tutti i calcolatori costruiti finora *hanno la medesima capacità di calcolo*; in altri termini, se si trova un algoritmo per risolvere un certo problema che sia eseguibile da un calcolatore *X*, si può essere certi di poter "programmare" il medesimo algoritmo, e quindi risolvere lo stesso problema, me-

diante un calcolatore *Y*: la semplicissima macchina di Turing descritta nella sezione precedente è in grado di risolvere gli stessi problemi che possono essere affrontati mediante un costosissimo calcolatore moderno. Una breve riflessione dimostra che questa affermazione non è così incredibile: basta infatti constatare che il contenuto dell'intera memoria di un qualsiasi calcolatore reale può essere "trasportato" nel nastro della macchina di Turing e che ogni istruzione del calcolatore può essere simulata da un opportuno numero di operazioni elementari (una sorta di sottoprogramma) della medesima.
Lo stesso vale per altri modelli formali come le grammatiche e, soprattutto, per ogni altro modello di calcolo, astratto o reale, che si possa inventare in futuro. Quest'ultima affermazione, ovviamente non può essere enunciata come un teorema poiché non si possono definire i modelli di calcolo ancora non inventati; essa è però supportata da tutta la storia pregressa e da un'analisi attenta del concetto generale di "operazione elementare". Affascinante, da un punto di vista storico, è il fatto che questa enunciazione sia dovuta ad Alonso Church e ad altri precursori della teoria della computazione, tra cui lo stesso Turing, che la formularono negli anni 30 del XX secolo, ben prima che vedesse la luce il moderno calcolatore elettronico. Per questo motivo, questo fondamentale pilastro della teoria della computazione è anche noto come *Tesi di Church*[1].
Armati di questo fondamentale risultato possiamo dunque tornare alla domanda di partenza, riformulandola in maniera matematicamente più precisa ma non per questo meno generale:
"*Quanti e quali problemi sono risolvibili mediante macchine di Turing*?"
La risposta comporta una sostanziale limitazione nella potenza del calcolo automatico: a dispetto di un'infinità di algoritmi già noti per la soluzione di problemi di ogni tipo e di tanti altri che verranno elaborati in futuro, possiamo affermare che *una altrettanto grande - anzi, una ben "maggiore - infinità" di problemi non è risolvibile mediante calcolo automatico*. Nel presente articolo ci limitiamo ad affermare questo fondamentale risultato; nei paragrafi seguenti esamineremo alcuni tipici problemi che non possono essere risolti algoritmicamente, e ne sottolineeremo l'impatto sia pratico che concettuale. Introdurremo invece le principali tecniche matematiche che portano a dimostrare questo ed altri risultati della teoria della computazione nell'articolo seguente di questa serie. Anche in questo caso evidenzieremo gli strettissimi rapporti e comunanze tra i vari filoni dell'evoluzione del pensiero umano: dalla matematica alla filosofia, all'informatica.

Impatto pratico dei limiti teorici alla calcolabilità

Storicamente, il problema più classico non calcolabile, ossia non risolvibile mediante un algoritmo, riguarda proprio una proprietà del calcolo automatico, ossia la sua terminazione. In termini teorici esso è noto come "*il problema dell'halt della macchina di Turing*: data una macchina M e un dato di partenza x, scritto sul nastro di M, la computazione di M avrà prima o poi termine?". Grazie alla tesi di Church questa domanda è del tutto equivalente alla seguente, dal "sapore" decisamente più pratico: "Ho scritto un programma in C e ne ho lanciato l'esecuzione fornendo in ingresso certi dati; dopo diversi secondi/minuti/ore il programma non mi ha ancora dato risposta: ne devo concludere che "è andato in loop" e quindi spegnere il calcolatore o abortire l'esecuzione o devo pazientare ulteriormente sperando che in un prossimo futuro giunga la risposta?".
Molti di noi, anche se non "informatici professionisti" saranno probabilmente passati almeno una volta attraverso questa frustrante esperienza ed avranno perciò constatato quanto grave sia la mancanza di un "messaggio diagnostico" da parte del compilatore che ci avverta a priori che se proviamo ad eseguire il programma testé scritto con certi dati esso è destinato a non terminare mai. Perché invece, il compilatore ci avverte, per esempio, che nel nostro programma abbiamo dimenticato di chiudere una parentesi e magari ci suggerisce anche dove e come eseguire la correzione? Perché il compilatore è a sua volta un algoritmo e il problema di stabilire se un'espressione è ben parentetizzata è decidibile[2], mentre

[1] Il lettore non si lasci fuorviare da certe affermazioni apparse in pubblicazioni a carattere divulgativo tendenti a sostenere che "la tesi di Church è ormai superata": si tratta di affermazioni ad effetto del tutto prive di rigore teorico.

[2] Il termine "decidibile" è sinonimo del termine "calcolabile" laddove il problema da risolvere sia espresso in termini booleani, ossia abbia una soluzione binaria.

il problema della terminazione dell'esecuzione di un algoritmo non lo è.
Come dicevamo, molti altri problemi sono sfortunatamente non calcolabili. Alcuni tipici esempi nell'ambito della programmazione sono i seguenti:

- L'occorrenza di una divisione per 0: durante l'esecuzione del mio programma, potrà capitare che si verifichi un tale errore?
- Accesso ad un array con un indice il cui valore è fuori dai limiti imposti dalla dichiarazione del medesimo array.
- Accesso ad una variabile non inizializzata, ossia cui non è stato ancora assegnato alcun valore durante l'esecuzione.

Chiunque abbia un po' di esperienza di programmazione sa bene che errori del genere sono molto frequenti e spesso hanno conseguenze anche gravissime (e gli hackers ne approfittano!). Perciò, l'impossibilità di individuarli mediante un opportuno algoritmo costituisce un grave impedimento alla produzione di software di qualità e pone maggiori responsabilità sulle spalle del programmatore.
In ambito matematico è ben noto che alcune equazioni ammettono soluzione mentre altre no; in taluni casi però esistono algoritmi per il calcolo della soluzione di un'equazione mentre in altri casi non ne esistono. Un esempio classico in tal senso è il famoso decimo problema di Hilbert, enunciato dal noto matematico all'inizio del XX secolo: "dato un polinomio a coefficienti interi in un numero qualsiasi di variabili, esistono radici intere del dato polinomio?". Solo alla fine degli anni 60 è stata dimostrata l'indecidibilità di questo problema: non esistono algoritmi per rispondere a questa domanda.
Si noti tuttavia che se ci si limita a considerare il caso di polinomi in una sola variabile, il problema diventa decidibile: è abbastanza facile costruire un algoritmo che, dato un polinomio in una variabile, stabilisca se ne esistono radici intere e, in caso positivo, le calcoli; questa circostanza ha una valenza generale: spesso accade che un problema non sia risolvibile in una sua formulazione generale ma lo diventi se opportunamente ristretto a casi particolari.
È necessario sottolineare che in questo contesto il termine "problema non risolvibile", non significa che non è possibile trovarne la soluzione in assoluto ma significa che non esistono algoritmi per la sua soluzione. Per esempio, in molti casi un'attenta analisi di un programma potrà far capire se esso è esposto al rischio della non terminazione o di un qualsiasi altro errore del tipo di quelli summenzionati. Questa analisi però, non sarà il risultato dell'esecuzione di un algoritmo, bensì il frutto di immaginazione, esperienza e, perché no, fortuna, doti queste, tipicamente umane e "non meccanizzabili"[3].

Impatto filosofico dei limiti teorici alla calcolabilità

Un importante settore applicativo dell'informatica è la dimostrazione automatica di teoremi. A dispetto del "suono decisamente matematico" di questo termine, esso ha notevoli risvolti pratici. Infatti, le formule matematiche altro non sono che una notazione precisa e astratta per descrivere problemi di vario tipo: così come un'equazione differenziale può descrivere la traiettoria di un missile e il problema della sua intercettazione, un'altra formula può esprimere il fatto che un impianto nucleare è sicuro, ossia garantisce dal rischio di esplosioni, perdite radioattive ecc. In questo contesto, perciò, dimostrare un teorema equivale a dimostrare la sicurezza di un impianto, la correttezza di un programma ecc.
A questo punto non dovrebbe sorprendere che la dimostrazione di teoremi è un altro problema non calcolabile: non esistono algoritmi generali per decidere se da certe ipotesi si possono trarre certe tesi. Questa nuova constatazione di limite alle possibilità del calcolo automatico ha risvolti non solo pratici, ma anche filosofici. Essa ci suggerisce infatti il passaggio dall'investigazione dei limiti del calcolo automatico all'investigazione dei limiti dello stesso ragionamento umano. Senza pretesa di sviluppare un affascinante ma complesso e delicato argomento in modo approfondito, cercheremo in questa sezione di far capire come da certi risultati dell'informatica teorica si possa risalire ai grandi temi filosofici che da sempre hanno sfidato il pensiero umano.
Come accennato in precedenza, è naturale la tendenza a descrivere il ragionamento umano in forma matematica: il tipico sillogismo aristotelico è un esempio di come il concetto di teorema formalizzi il ricavare una nuova verità a partire da altre verità assunte come ipotesi. In generale chiunque è disposto ad accettare come verità un

[3] Fatte salve alcune considerazioni proposte nella sezione seguente.

teorema ben dimostrato. La dimostrazione di teoremi risulta perciò un importante strumento per ricavare "verità", qualsiasi sia il contesto in cui tali verità sono ricercate (l'aritmetica, le operazioni finanziarie, il controllo di impianti, strategie belliche ecc.). Se però è del tutto accettabile - sotto opportune ipotesi - che un teorema possa essere assunto come verità, il problema inverso relativo alla "ricerca di verità" mediante dimostrazione di teoremi è tutt'altro che scontato. Infatti, sempre nella prima metà del XX secolo, Kurt Goedel ha dimostrato i suoi fondamentali *teoremi di incompletezza*, che, in qualche maniera, sintetizzano millenni di studi sulla capacità e sui limiti del ragionamento umano. L'essenza filosofica dei teoremi di Goedel potrebbe essere formulata nel modo seguente:

> *"Qualsiasi formalizzazione del ragionamento umano non è in grado di catturare in modo completo tutte le verità che sono conseguenze di certe ipotesi"*

Dalla filosofia alla religione il passo è breve e così più d'uno, incluso lo stesso Goedel in tarda età, ha ritenuto di poter ricavare da questa constatazione una "dimostrazione" dell'esistenza di un'Entità Superiore, capace di catturare tutte quelle verità che sfuggono ai limiti umani; ciò costituisce un errore logico altrettanto grave dell'affermazione opposta[4].

Prima di chiudere questo articolo dedichiamo un breve cenno al settore dell'Intelligenza Artificiale. Questo termine, spesso abusato, include studi ed obiettivi che coprono aspetti di grande valenza pratica, come la robotica, e di profondo studio filosofico, elemento accomunante essendo il desiderio di ottenere dalla macchina risultati comparabili o superiori a quelli ottenuti dall'intelligenza umana (per esempio, nelle partite a scacchi, nella traduzione del linguaggio naturale ecc.). A prima vista, quanto affermato in precedenza potrebbe portare alla conclusione che necessariamente "il cervello umano non è un semplice esecutore di algoritmi" e quindi il termine "intelligenza" include capacità diverse da ciò che si può affrontare in modo algoritmico. In realtà, però, è ammissibile anche l'ipotesi opposta, sposata da diversi pensatori, tra cui lo stesso Turing. L'apparente contraddizione potrebbe essere spiegata con l'ipotesi che, quando noi risolviamo un problema non calcolabile stiamo certamente eseguendo un algoritmo, ma un algoritmo che non ci garantisce di giungere al risultato cercato: in un certo senso, stiamo "tentando un strada algoritmica", ma diversamente da quando risolviamo un problema calcolabile, non sappiamo se si tratti della strada giusta né se questa ci porterà mai ad un risultato. Alcune tecniche illustrate nell'articolo successivo potranno forse aiutarci a capire meglio questa congettura.

[4] Scusandoci per l'inevitabile banalizzazione di un tema così profondo e delicato, commette un errore logico chi ritiene di aver dimostrato un'affermazione per il fatto di non trovare altro modo di spiegare un fenomeno, così come chi dichiara falsa un'affermazione per il fatto che non ne esista dimostrazione.

Bibliografia

Essendo impossibile rendere completa giustizia ai tanti "pionieri" dell'informatica (che a nostro modo di vedere includono matematici e filosofi greci e arabi di millenni trascorsi) ci limitiamo qui a citare alcune pietre miliari della teoria della computabilità ([2] e [4]), della linguistica matematica [1]) e della logica [3].

[1] Chomsky N.: Three Models for the Description of Languages. *IRE Transactions on Information Theory*, Vol. 2, n. 3, 1956, p. 113-124.

[2] Church A.: An Unsolvable Problem of Elementary Number Theory. *American Journal of Mathematics*, Vol. 58, 1936, p. 345-363.

[3] Goedel K.: *On Undecidable Propositions of Formal Mathematical Systems*. Princeton University Press, 1934.

[4] Turing A.: On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings London Mathematical Society*, Vol. 42, p. 230-265 e Vol. 43, p. 544-546, 1936-1937.

Dino Mandrioli è professore ordinario di Informatica Teorica presso il Politecnico di Milano. I suoi interessi di ricerca sono principalmente nei settori dell'informatica teorica, dell'ingegneria del software e dei sistemi critici in tempo reale. Ha pubblicato oltre 80 articoli scientifici su riviste ed atti di convegni internazionali. È coautore di vari libri, fra cui *Theoretical Foundations of Computer Science* (J. Wiley & Sons), *Fundamentals of Software Engineering* (Prentice-Hall), *The Art and Craft of Computing* (Addison Wesley). Dino Mandrioli è stato membro del Program Committee di diverse conferenze internazionali, Associate Editor di diverse riviste internazionali, program co-chairman della conferenza Formal Methods 2003.