# Informatica e Civiltà: Logica, Tecnologia e Sapienza

Egon Börger

## Il doppio ruolo della logica tra sapienza e tecnologia

**boerger@di.unipi.it**
**Dipartimento di Informatica**
**Università di Pisa**

Pisa 9.12.2003

# Il doppio ruolo della logica

La logica è

A. all'origine della tecnologia informatica

avendone fornito due concetti di base

- leggi del pensiero
- algoritmo universale

B. uno strumento per usare la tecnologia informatica con sapienza

fornendo

- una corretta prospettiva della portata di questa tecnologia
- dei metodi per rimanerne padroni

# A. La logica all'origine della tecnologia informatica

- l'idea di **leggi del pensiero universalmente validi**
  sostenuta per primo da Platone contro le tesi dei sofisti
  - analizzata da Aristotele per 'enunciati' ( $\lambda\acute{o}\gamma o\sigma\ \alpha\pi o\varphi\alpha\nu\tau\iota\kappa\acute{o}\sigma$, frase che è o vera o falsa) ed implementata con il primo **calcolo deduttivo**
    - per 'syllogismi' (Analytica Priora)
    - per 'fallacie' come nozione complementare della nozione di legge logica (Elenchi Sofistici e Primi Analitici)
  - analizzata anche dagli stoici per la formale 'logica proposizionale'
    invenzione del metodo delle 'tavole di verità' !
- l'idea di **algoritmi** (regola di calcolo), realizzata in tre fasi:
  1. sviluppo di **algoritmi concreti** (da Euclide 400 a.C. in poi)
  2. l' idea di un **algoritmo universale** (Raimundus Lullus, ca. 1300)
  3. **definizione matematica** del concetto di algoritmo (1914-1935/6) —che scopre inaspettate potenzialità e limitazioni

# A.1 Sviluppo di algoritmi concreti

- Euclide (Grecia, tra 400 e 300 a.C.): algoritmo per calcolare il massimo comun denominatore, il primo algoritmo importante a noi noto
- ...
- Al-Chwarizmi (ca. 800, Persia), matematico arabo da cui deriva il nome 'algoritmo': algoritmi per le 4 operazioni numeriche elementari in notazione decimale
- ...

Lo sviluppo e l'analisi di

algoritmi efficienti

è anche oggi un compito fondamentale dell'informatica

# A.2 L'idea di un algoritmo universale (ars magna)

procedimento generale su base combinatoria per trovare tutte le verità

- idea formulata da **Raimundus Lullus**, 1235–1315/6 (Mallorca)
  personaggio interessante e molto influente: guerriero, professore a Parigi e Montpellier, frate Francescano, martire.

  Lullus sviluppa la ars magna come metodo per confutare i Saraceni!
- l'idea prende corpo nel famoso **calculemus**
  - formulato da G. W. Leibniz (1646-1716, Germania) come metodo per dirimere le dispute scientifiche con le regole di un calcolo
  - ripreso dal programma di Hilbert per proteggere la matematica contro i paradossi della teoria degli insiemi (G. Cantor, B. Russell)

NB. Leibniz costruì la **prima macchina calcolatrice** (Landesbibliothek Hannover), 'special purpose' per le quattro operazioni aritmetiche elementari dove divisione e moltiplicazione sono ricondotte ad addizione e sottrazione

# Leibniz: precisazione di 'calcolo' e 'linguaggio universale'

Leibniz comprende la necessità, per un tale 'calculus ratiocinator'

- di una **characteristica universalis**: un linguaggio logico formale generale

  Per ottenere l'universalità del linguaggio Leibniz inventa la codifica $n$-aria (per ogni $n > 1$) di qualsiasi sequenza finita di simboli!
- della distinzione tra
  - **ars inveniendi**: procedimento effettivo di enumerazione
  - **ars iudicandi**: procedimento effettivo di decisione

Un tale linguaggio logico 'formale' viene realizzato molto piu tardi

- per la logica proposizionale da G. Boole (1815-1864, Inghilterra) con ars iudicandi
- per la logica dei predicati da G. Frege (1848-1925, Germania) con ars inveniendi

  dimostrata tale da Gödel (teorema di completezza, 1930) solvendo il problema formulato nel libro di Hilbert-Ackermann 1928

# A.3 Definizione matematica di 'algoritmo'

- **macchina di Turing** (A. Turing, 1912-1954, Inghilterra)
  - numerose varianti dimostrabilmente equivalenti
    A. Thue, E. Post, A. Church, K. Gödel, J. Herbrand, A. Markov, H. Wang, ...
    Giustificazione epistemologica della appropriatezza della definizione tramite la **Tesi di Church-Turing**
- scoperta di due fenomeni sconvolgenti che derivano dalla definizione matematica del concetto di algoritmo
  - **potenzialità** inaspettate: macchine programmabili universali
  - **limiti** intrinseci:
    - indecidibilità (algoritmica)
    - incompletezza (deduttiva)

# A.3.1 Potenzialità sconvolgenti di algoritmi

- scoperta del **concetto di macchina programmabile** nella forma di 'macchina di Turing universale' (1935)

  Era rivoluzionario pensare che si possono costruire general purpose machines capaci di sostituire la miriade di sofisticatissime macchine special purpose (come le costruiva con successo la IBM—Intern. Bureau Machines!— di quei tempi!)

  - prima implementazione di una macchina programmabile: **Z3 di K. Zuse** (1941, Germania)

    basata sul **Plankalkül**: il primo linguaggio di programmazione derivato dal linguaggio logico di Hilbert-Ackermann 1928 (che è una characteristica universalis nel senso di Leibniz)

  - la programmabilità diventa parte integrale della **architettura von Neumann** (1903 Budapest-1957 Princeton): EDVAC Report 1945

    'real'izza le idee spiegate da Turing a Princeton 1936-1938: 'program as data in memory' e 'instruction execution by ALU'

# A.3.2 Limiti intrinseci sconvolgenti di algoritmi

Sulla base della precisazione matematica del concetto intuitivo di algoritmo si riesce a dare una **prova matematica** di suoi limiti

- K. Gödel (1906 Brno-1978 Princeton) prova **l'incompletezza di metodi deduttivi** (1931)
  Come nel Zauberlehrling di Goethe l'apprendista stregone dà inizio alla magia ma non è in grado di portarla a termine, così finisce il programma di Hilbert di ridurre il sapere matematico a tecnologia di calcolo (deduzioni da un unico sistema universale di assiomi)
- Turing (1935/36) dimostra **l'insolubilità algoritmica**
  - di Hilbert's 'Entscheidungsproblem' (Hilbert-Ackermann 1928): segna la fine del sogno del calculemus di Leibniz
  - di qualsiasi proprietà non banale di algoritmi ('Halting Problem', Teorema di Rice 1953)

Le prove usano la tecnica di diagonalizzazione della teoria degli insiemi (G. Cantor, 1845 Pietroburgo–1918 Halle)

# Però: la fine del logicismo non è la fine della logica

La tecnologia deve ritornare nei ranghi in cui già l'aveva posta Aristotele 2000 anni fa:

La **tecnologia** non è sapere ma è il suo prodotto

Il **sapere** è la scienza dei principi primi

che si fonda sulle capacità creative della mente umana
e cioè dell'intuizione rispetto al calcolo ed alla deduzione

# B. La logica come strumento

## per usare la tecnologia informatica con sapienza

La logica

- **epistemologicamente** parlando
  aiuta a mettere le tecnologie informatiche nella giusta luce
  - deduzione versus intuizione
- **tecnicamente** parlando
  - fornisce un unifying foundational framework per i concetti di base dell'informatica
  - aiuta a domare la complessità di tecnologie informatiche attraverso l'astrazione
- **pragmaticamente** parlando
  insegna a non frenare il libero gioco delle idee

# B.1 Mettere le tecnologie informatiche nella giusta luce

La logica offre una **corretta valutazione delle capacità di general purpose machines** (algoritmi universali)—per non cadere vittima di convinzioni su una presunta onnipotenza dei computer.

In particolare, nel far vedere le potenzialità ed i limiti interni di formalismi deduttivi, la logica aiuta a non perdere di vista la

differenza tra metodi computerizzati ('formali') e metodi creativi

- critica di Poincarè al programma logicistico di Russell-Whitehead:
  Una macchina che prende assiomi per produrre teoremi è 'come la legendaria macchina di Chicago dove i maiali entrano vivi da una parte ed escono trasformati in prosciutti e salsicce'
- considerazione di von Neumann nel 1927, otto anni prima della scoperta della insolubilità algoritmica del Entscheidungsproblem:
  La impossibilità di una ars iudicandi per la logica dei predicati è la 'conditio sine qua non' per dare senso all'attività matematica esercitata con i metodi euristici odierni

# B.2 Quadro fondazionale di concetti di base dell'informatica

La logica fornisce un quadro rigoroso uniforme per un grande numero di concetti fondamentali della tecnologia e della scienza informatica

- **sintassi** e **semantica** e loro relazioni
  linguaggio formale, struttura, sostituzione, interpretazione, ...
- sistemi di **tipi** e gerarchie di classi
- concetti di **computazione** e misure della loro **complessità**
  - macchine: a stati finiti (FSM), a pila, virtuali (ASM), ...
  - forme di computazione
    procedure (call by name vs call by value), ricorsioni (definizioni esplicite vs definizioni implicite), term rewriting, ...
  - ...
- concetti e metodi di **strutturazione** e di **composizione**
- concetti e metodi di **verifica** e loro classificazione
- ...

# B.3 Domare la complessità di tecnologie informatiche

Le **potenzialità di astrazione** del pensiero logico

- da particolarità del dominio di applicazione, di linguaggi di programmazione (strutture di controllo e di dati), di piattaforme

aiutano a **dominare la complessità** per

mantenere comprensione e controllo umano di sistemi informatici

attraverso lo sviluppo di una gerarchia di modelli logici astratti (ASM), che porta **in maniera affidabile**, a passi successivi,

dal livello del dominio dell'applicazione reale fino al codice

- il **modello di base** (scritto in termini applicativi 'human-centric') collega il sistema con il mondo reale
- i modelli di livelli successivi sono connessi tramite **raffinamenti controllabili** che documentano le principali decisioni del disegno

Simonyi, padre di wysiwyg MS Word: show 'software intent'

# B.4 Aiutare il libero gioco e sviluppo delle idee

La storia dei concetti sottostanti la tecnologia informatica fa capire

il potere delle idee e l'impossibilità di prevedere dove portano

- dal metodo di Lullus per confutare i Saraceni— allo sviluppo dell'idea di Leibniz di una characteristica universalis e dell'invenzione della rappresentazione binaria di tutte le informazioni!
- dalla diagonalizzazione di Cantor—a stabilire limiti inferiori di complessità di algoritmi o il grado di sicurezza di chiavi criptografiche!
- dall'algebra della logica di Boole—allo sviluppo di circuiti elettronici!
- dalle regole di deduzione di Frege—a prove computerizzate!
- dalla teoria dei tipi di Russell-Whitehead per evitare i paradossi della teoria degli insiemi—a tipare i moderni linguaggi di programmazione per poter scoprire e prevenire degli errori di programmazione!
- dai sistemi Thue o dalle funzioni ricorsive di Gödel—allo sviluppo di strumenti per l'implementazione di linguaggi di programmazione!

# Due riferimenti bibliografici

Martin Davis
The Universal Computer - The Road from Leibniz to Turing
W.W.Norton, New York 2000

Trad. Italiana: Il calcolatore universale
Adelphi 2003
http://www.newton.rcs.it/Recensioni/Libri

Egon Börger and Robert Stärk
Abstract State Machines - A Method for High-Level System Design and Analysis
Springer, Berlin etc. 2003
http://www.di.unipi.it/AsmBook/