Estratto da

M. Barra e A. Zanardo (a cura di), *Atti degli incontri di logica matematica* Volume 5, Roma 6-9 aprile 1988.

Disponibile in rete su http://www.ailalogica.it

INFORMATICA, LOGICA, DIDATTICA

A. BERTONI, G. MAURI
Dip. Scienze Informazione – Università di Milano

ABSTRACT

In this paper we briefly discuss some aspects of the interaction among logic and computer science. In order to better understand the problems posed by the computer science, some considerations about its social dimension in Italy today and about its technological aspects are given. Then some of the positions in the debate about the mathematical contents of the curricula in computer science are critically discussed in a methodological perspective. The last two sections treat the relationships among logic and research in computer science: on the one hand, computer science uses some logical tools, on the other gives new notivations to the research in logic. In particular, the example of interactive proof systems, a stimulating research topic in which motivations, tools and techniques both from logic and computer science are put together and a quantitative notion of knowledge is given, are treated in detail.

INTRODUZIONE

In questo lavoro cercheremo di fissare qualche idea sul rapporto tra informatica e logica, anche rispetto alle esigenze della didattica. Non si può tuttavia prescindere da una breve disamina di altri aspetti dell'informatica, disciplina complessa che non può essere ridotta a un capitolo della matematica, considerandola come una disciplina puramente formale, ma nemmeno ad un insieme di "tecniche realizzative".Essa invece si muove su più piani, presenta diverse "facce" che vanno tutte tenute presenti per una reale comprensione.
Il primo piano è quello sociale. Soprattutto da quando, grazie al "personal computer", l'informatica è uscita dai grandi centri di calcolo ed è stata messa alla portata di tutti, essa ha letteralmente invaso la vita quotidiana anche dei non addetti ai lavori: giornalisti, impiegati, insegnanti, oltre ad essere entrata in migliaia di piccole aziende. Di conseguenza, è diventata argomento di servizi giornalistici, libri divulgativi, dibattiti sulla necessità di insegnarla in

tutte le scuole e così via; dedichiamo il primo paragrafo alla presentazione di una serie di dati che diano un ordine di grandezza del fenomeno in Italia.
Questa esplosione, si è detto, è legata ad un preciso evento tecnologico, e questo ci porta al secondo piano, quello della tecnologia che ha fortemente influenzato lo sviluppo dell'informatica, spesso determinando scelte precise anche sul piano dei concetti. Nel secondo paragrafo discutiamo di recenti eventi tecnologici che sicuramente indirizzeranno le linee di ricerca scientifica; una lucida e generale esposizione di questi argomenti si trova in [1].
Un ampio dibattito si è sviluppato intorno ai contenuti matematici da introdurre negli insegnamenti di Scienze dell'Informazione [2], [3], [4]; in questo lavoro si accenna al problema su un piano storico, ponendo in evidenza in particolare alcuni aspetti metodologici.
Gli ultimi due paragrafi discutono attraverso esempi il rapporto tra la logica e la ricerca informatica: dopo aver presentato una serie di esempi di applicazione a problematiche informatiche di metodi sviluppati in ambito logico, si cerca di evidenziare come le richieste e le motivazioni informatiche possano dare luogo a nuovi sviluppi di ricerca. Vengono infine trattati in maggior dettaglio i Sistemi di Dimostrazione Interattivi, un argomento che, pur basato su idee motivate da ricerche sui fondamenti della matematica, è stato recentemente proposto in ambito di informatica teorica; riteniamo questo argomento particolarmente interessante sul piano epistemologico perché propone una nozione "quantitativa" di conoscenza acquisita tramite comunicazione.

1. Il fenomeno informatica

Per avere un ordine di grandezza dell'Informatica come fenomeno sociale, consideriamo alcuni dati relativi alla situazione italiana [5].
Quantitativamente, la spesa complessiva annua per hardware e servizi si aggira intorno ai 10.000 miliardi di lire, con una crescita annua superiore al 30%; oltre alle aziende multinazionali ben note, operano come costruttrici una ventina di aziende nazionali e come produttrici di software circa 3.000 aziende di cui un migliaio dotate di organizzazione industriale.
In attività informatiche risultano occupate 60.000 persone nei settori fornitori di apparecchiature, 40.000 nei settori fornitori software e 180.000 nelle aziende utenti; va rilevato che questo tipo di personale richiede buona qualificazione.
Per estrapolazione, è ragionevole pensare ad una richiesta di occupazione annua di 20.000

nuovi addetti, di cui 4.000 laureati. In aggiunta, un fenomeno non trascurabile è la nascita di nuove occupazioni, quali progettisti di reti, tecnici di automazione d'ufficio, esperti CAD-CAM, progettisti di sistemi esperti e così via.

2. Tecnologia e ricerca in informatica

Per capire le possibili linee di evoluzione della ricerca in informatica, non è possibile prescindere da alcune considerazioni tecnologiche. Come è ben noto, negli ultimi 20 anni si sono avuti considerevoli avanzamenti nella microelettronica. Per quanto riguarda i circuiti ad altissima integrazione, il numero di transistors per piastra è raddoppiato ogni 2 anni (almeno fino al 1986), mentre il costo di memorizzazione dell'informazione è sceso con un tasso analogo. Il passaggio dalla tecnologia n-MOS alla C-MOS comporta una differenza qualitativa: la dissipazione termica non è più proporzionale all'area, ma al volume di calcolo.
Accenniamo qui brevemente a due fenomeni che si sono di conseguenza creati. Un fenomeno caratteristico di questo periodo è la cosiddetta "spirale tecnologica": applicazioni complesse creano la necessità di circuiti complessi, di conseguenza risultano necessari strumenti di progettazione informatici complessi, che comportano applicazioni complesse. Questo comporta la nascita di nuove discipline (esempio la Progettazione con l'Aiuto del Calcolatore (CAD)), nuove linee di ricerca, nuove figure di ricercatori.
Un'altra nota conseguenza è il grande interesse di cui sono oggetto le problematiche relative alle architetture di calcolo parallele, che produce e produrrà nuove e differenti linee di ricerca.
La architettura di calcolo che ha accompagnato lo sviluppo informatico per più di un ventennio è la cosiddetta "architettura di von Neumann": una memoria, una unità preposta al trattamento dei dati, una unità di controllo preposta al trattamento di speciali dati detti programmi, il tutto connesso da una linea di comunicazione o "bus". In sintesi, possiamo dire che questa architettura è caratterizzata da un trattamento sia dei dati che del controllo "trasparente" all'utente, che può unicamente preoccuparsi dei problemi di programmazione: questo ha permesso la costruzione di un edificio di considerevole interesse e generalità che va sotto il nome di programmazione sequenziale. Ora, l'attuale tecnologia mette a disposizione piastre con più di un milione di transistors, che potebbero commutare tutti in parallelo: di qui il concreto interesse all'utilizzo di tale potenzialità. Questo punto di vista richiede di abbandonare il "giardino dell'Eden" creato dalla macchina di Von Neumann, per ritrovarsi alle prese con tutti i problemi che riassumiamo in uno slogan: mancanza di trasparenza tecnologica.

In conclusione, da un lato i problemi proposti in ambito della programmazione richiedono una struttura informatica sempre più dinamica, eterogenea, sequenziale ed affidabile, dall'altro le macchine saranno dotate di una struttura fisica statica, ripetitiva, altamente parallela e localmente difettosa: la necessità di coprire lo spazio che si allarga sempre più tra la struttura informatica e la struttura fisica determinerà i principali temi della ricerca in scienze dell'informazione.

3. Informatica e didattica

Un aspetto importante della rilevanza sociale dell'informatica è quello della sua presenza nella scuola. E' evidente che l'informatica pone la scuola di fronte a nuove esigenze e nuove domande di formazione, ma offre anche nuove possibilità per rendere più efficace l'intero processo educativo. Non ci interessa qui parlare dell'uso dei calcolatori per la didattica, che ha visto alternarsi negli ultimi trent'anni momenti di entusiasmi eccessivi e momenti di profonda delusione per la sostanziale mancanza di risultati concreti soprattutto nel settore del CAI (Didattica assistita da elaboratore), e che solo recentemente sembra aver imboccato strade meno velleitarie e più realistiche. Vogliamo piuttosto discutere brevemente il problema della didattica dell'informatica e della sua integrazione nel curricolum scolastico ai vari livelli.

Negli anni recenti si è molto parlato, e spesso a sproposito, di alfabetizzazione informatica, dando a questo termine i significati più disparati, dall'imparare a programmare in Basic fino a studiare la struttura interna della macchina o a conoscere la teoria degli automi. Difetto comune a tutti questi approcci é quello di concentrare l'attenzione più sui contenuti che sui metodi, e per di più su contenuti troppo specifici o troppo legati ad una particolare fase di sviluppo tecnologico. Una analoga situazione si è avuta anche nella discussione sui curricula per l'insegnamento dell'informatica a livello universitario. Ad esempio, nella proposta di curriculum avanzata dall'ACM nel 1978 [3] è possibile evidenziare una diminuzione dei corsi di base di matematica (sia in termini numerici che di loro propedeuticità rispetto a quelli di programmazione) rispetto al precedente curriculum del 1968 [2]. La motivazione chiave per questa scelta stava nella identificazione dell'informatica con la programmazione, tipica di quegli anni. Viceversa, crediamo che proprio la rapida obsolescenza delle tecnologie e la ricerca di metodi formali per lo sviluppo del software evidenzi la necessità di puntare su solide conoscenze di base, il più possibile indipendenti dalla tecnologia, e su aspetti metodologici più che di contenuto.

Tra le attitudini di tipo di metodologico che si sono rivelate importanti in informatica, la

capacità di astrazione è probabilmente quella di maggior portata. Astrazione significa prescindere da dettagli inessenziali, ignorando selettivamente la struttura e collocandosi in un ambito "ideale" o "virtuale" più semplice rispetto al mondo reale con cui si ha a che fare. Non sarebbe possibile "programmare in grande", scrivere programmi (ovviamente corretti) per risolvere problemi complicati, senza passare attraverso diversi livelli di astrazione: tutta l'ingegneria del software si regge sulla idea di astrazione. E lo stesso si può dire dei linguaggi di programmazione evoluti, che fanno uso di istruzioni considerate primitive per una macchina astratta o virtuale ben diversa dalla macchina reale che poi eseguirà i programmi.

Una seconda capacità importante è la attitudine a risolvere i problemi "effettivamente", cioè in termini algoritmici. Questa capacità, tipica dell'atteggiamento costruttivo in matematica, è stata estesa nel settore informatico anche ad ambiti non tradizionali, ed è stata arricchita con l'apporto di una specifica attenzione alla complessità computazionale degli algoritmi prima trascurata.

A livello di specifici contenuti tecnici utili alla formazione informatica, vanno segnalate in particolare le discipline che vanno sotto il nome di "matematiche discrete". Fra di esse, è stata da più parti segnalata l'importanza di un corso di logica elementare, da aggiungere ai corsi di base di matematica, e l'integrazione di tecniche combinatorie nei tradizionali corsi di calcolo differenziale.

4. Informatica e logica

Non è facile definire, anche se per sommi capi, le interazioni reciproche tra informatica e logica. Da un lato, l'esperienza di questi ultimi decenni ha posto in rilievo l'importanza che metodi e strumenti sviluppati in ambito logico possono assumere in informatica. Il riferimento, per certi versi scontato, alla teoria della ricorsività, nata da problematiche della logica matematica e divenuta una delle basi concettuali dell'informatica, può essere fuorviante, perchè ha carattere così generale da nascondere le richieste tecnologiche e pratiche che stanno alla base dell'uso di strumenti logici in informatica. Lo studio della correttezza dei programmi (cioè stabilire se un dato programma calcola effettivamente la funzione voluta) può essere un esempio chiarificatore a questo proposito. La teoria della ricorsività classica, orientata a dare soluzioni "in negativo", mostra che tale problema è in generale indecidibile. Viceversa, la necessità, nell'ambito informatico, di dare una soluzione "in positivo" a questo problema, divenuto rilevante per quanto riguarda i costi della produzione del software, ha portato allo sviluppo di metodologie, basate inizialmente su idee derivate dal calcolo dei predicati, che si pongono come obiettivo la possibilità di costruire direttamente programmi corretti [6].

Poichè la verifica formale di correttezza ha senso solo se si dispone di una specifica formale del problema da risolvere, è chiaro che l'attenzione viene spostata sugli strumenti necessari per fornire questa specifica. Si sono quindi introdotti appositi linguaggi di specifica che a loro volta si appoggiano a strumenti sviluppati nell'ambito dell'algebra universale. Questi esempi mostrano che sono motivazioni di ordine pratico che filtrano e determinano l'interesse in informatica di particolari strumenti logici.
Analoghe situazioni si presentano per l'area della sintesi dei programmi, che ha attinto ad idee di teoria della dimostrazione, e per i linguaggi logici e funzionali. Ad esempio, l'interesse suscitato da linguaggi come Prolog e dalla programmazione logica [7] è probabilmente dovuto al buon compromesso ottenuto tra la possibilità di specificare problemi e darne una soluzione algoritmica (non eccessivamente inefficiente) con tecniche di deduzione automatica.
Se su un versante l'informatica ha utilizzato strumenti presi dalla logica in base alle proprie esigenze, spesso determinate da precise richieste tecnologiche, sull'altro ha contribuito a dare nuove direzioni a ricerche inizialmente nate in ambito logico, ed a fornire motivazioni per lo sviluppo di nuove aree di ricerca in logica.
Un esempio di area di ricerca ravvivata da motivazioni provenienti dall'informatica è quella della logica temporale, di cui sono state proposte nuove varianti utilizzate per specificare proprietà utili per l'analisi di programmi concorrenti [8].
Un altro esempio significativo è quello della teoria della complessità computazionale, nata da una rilettura in chiave informatica di problemi già trattati dalla logica con la teoria della ricorsività. Una domanda centrale in ricorsività è se un certo problema ammetta o meno algoritmi di soluzione. Ora, in caso di risposta positiva, la necessità di "eseguire effettivamente" l'algoritmo pone un'altra questione, e cioè quanto costa questa esecuzione, in termini di risorse come il tempo di calcolo o la quantità di memoria occorrente.
Si scopre così che ci sono problemi decidibili per cui l'algoritmo più veloce di soluzione richiederebbe già per dati esprimibili con pochi caratteri miliardi di anni di calcoli, anche con macchine velocissime. Si è così arrivati da un lato a distinguere i problemi computazionalmente facili (cioè risolubili in tempi ragionevoli) da quelli difficili [9] attraverso tecniche formali in molti casi ispirate a quelle della ricorsività, dall'altro a mettere a punto tecniche di una certa generalità per il progetto di algoritmi efficienti.

5. Un esempio: i Sistemi di Dimostrazione Interattivi

Un interessante esempio di sinergia tra logica e informatica, che, anche se informalmente, descriveremo un po' più in dettaglio, perchè presenta aspetti storici e epistemologici non

trascurabili, è quello dei Sistemi di Dimostrazione Interattivi (Interactive Proof Systems, IPS) a conoscenza 0 [10]. Essi mostrano la possibilità di convincere un interlocutore, con un grado di certezza grande quanto si vuole, di possedere la dimostrazione di un fatto matematico senza presentarla esplicitamente.
Le idee base di IPS sono storicamente motivate da problematiche di tipo fondazionale, quali la critica ai fondamenti della teoria della probabilità e la conseguente introduzione della nozione di casualità legata alla ricorsività dovuta a Kolmogorov [11], lo sviluppo di metodi non costruttivi per l'analisi dei problemi dovuto a Erdös [12], unitamente a tecniche di classificazione introdotte in teoria della complessità. Fra le motivazioni pratiche, la principale è legata alla crittografia a chiave pubblica: si tratta di dare al mittente A di un messaggio informazioni sufficienti per crittografarlo nel codice del destinatario B senza metterlo però in grado di decifrare altri messaggi destinati a B [13].

L'idea principale è immediatamente comprensibile a chi abbia un minimo di esperienza di matematica: dimostrare è in genere più difficile che verificare la dimostrazione.
Questa nozione è colta nella classe di complessità NP [9]; ad esempio il problema di dimostrare la soddisfattibilità di una data formula booleana (noto problema in NP) è difficile, poiché richiede di selezionare un opportuno assegnamento (che, soddisfacendo la formula, "testimonia" la dimostrazione) in un insieme di cardinalità esponenziale rispetto al numero di variabili nella formula. Se invece un "indovino" fornisce un assegnamento sostenendo che esso rende vera la formula, questa affermazione può essere verificata facilmente anche da una persona "incredula".
In situazioni di questo tipo si può allora distinguere tra un DIMOSTRATORE D di potenza computazionale alta, in grado di fornire una dimostrazione ω_x di x, ed un VERIFICATORE V di potenza limitata (tecnicamente: in tempo polinomiale probabilistico) in grado di controllare che ω_x è dimostrazione di x. Specificando un protocollo di interazione tra D e V si ottiene un Sistema di Dimostrazione Interattivo.
Diremo che un linguaggio L ammette un IPS se è possibile trovare un dimostratore D e un verificatore V tali che:
1) se $x \in L$ allora il sistema interattivo <D,V> accetta x con probabilità "vicina a 1" (completezza)

2) se $x \notin L$ allora per ogni dimostratore D' il sistema interattivo <D',V> respinge x con probabilità "vicina a 1" (validità).

Cerchiamo di chiarire i concetti sopra illustrati con un esempio: il problema "non

isomorfismo tra grafi". Ricordiamo che, dato N = {1,2,..,n}, un grafo con nodi N è descritto dalla coppia <N,E> dove E⊆NxN. Due grafi G_1=<N, E_1> e G_2=<N, E_2> sono detti "isomorfi" se esiste una funzione invertibile f: N → N tale che (x,y)∈ E_1 se e solo se (f(x),f(y))∈ E_2.

Il problema "non isomorfismo tra grafi" può essere così espresso:
Istanza: due grafi G_1=<V, E_1> e G_2=<V, E_2>
Questione: sono i due grafi G_1=<V, E_1> e G_2=<V, E_2> non isomorfi?

Il protocollo di un Sistema di Dimostrazione Interattivo che risolve il problema è il seguente:

(1) L'interazione viene iniziata dal verificatore V che:
- Sceglie a caso a∈ {1,2};
- Sceglie a caso una funzione invertibile f: N → N
- Costruisce il grafo H ottenuto da G_a ridenominando con f(x) ogni vertice x
- Chiede a D se H è isomorfo a G_1 o a G_2.

(2) Il dimostratore D risponde con b∈ {1,2} (b=1 significa che H è isomorfo a G_1, b=2 significa che H è isomorfo a G_2).

(3) Infine V confronta a con b e se a≠b si arresta concludendo che i due grafi sono isomorfi, per cui la risposta al problema è negativa (infatti H è per costruzione isomorfo a G_a, e dal dimostratore dichiarato isomorfo a G_b , e quindi per la proprietà transitiva G_a è isomorfo a G_b). Se viceversa vale a=b avvia un secondo round di interazione ripartendo da (1) e così via.

Dopo k round di interazione, la probabilità che il problema abbia risposta positiva è almeno $1-2^{-k}$, e quindi essa può essere resa arbitrariamente vicina a 1 iterando un numero sufficientemente alto di volte.

Il formalismo introdotto è basato sulla comunicazione tra D e V; poichè comunicazione significa scambio di conoscenza, è possibile introdurre una nozione quantitativa di conoscenza scambiata durante la comunicazione, basata sulla assunzione che la conoscenza è una nozione relativa ad uno specifico modello di calcolo con risorse specificate.
Questo implica l'introduzione di una distinzione tra informazione e conoscenza. Se A trasmette a B un messaggio, gli trasmette una informazione, ma non necessariamente una conoscenza. Ad esempio, se A trasmette una sequenza casuale di bit, non trasmette alcuna conoscenza, poichè anche B è in grado di generare una sequenza casuale di bit. Diremo allora che A trasmette conoscenza a B se e solo se il messaggio contiene l'output di una computazione che B non è in grado di eseguire.
Ritornando all'esempio precedente, si può concludere che il protocollo ci permette di ottenere una dimostrazione in cui il Verificatore acquisisce una conoscenza veramente limitata, indipendente dalla dimensione del grafo. Sono infatti sufficienti, ad esempio, 7 bits per acquisire una sicurezza almeno del 99% di correttezza della risposta (sia affermativa che negativa) pur non essendo stato esibito alcun "testimone" (cioé un isomorfismo) della dimostrazione stessa.

RIFERIMENTI BIBLIOGRAFICI

[1] J. Vuillemin, Nouvelles structures d'ordinateurs, in "VLSI: Algorithms and Architectures" (P. Bertolazzi and F. Luccio ed's), North-Holland, 1985, pp.137-151

[2] Curriculum 68: Recommendations for academic programs in computer science - A report of the ACM Curriculum Committee on Computer Science, Comm.ACM, vol.11, n.3, 1968, pp.151-197

[3] Curriculum 78: Recommendations for the undergraduate program in computer science - A report of the ACM Curriculum Committee on Computer Science, Comm.ACM, vol.22, n.3, 1979, pp.147-166

[4] A. Berztiss, A mathematically focused curriculum for computer science, Comm. ACM, vol.30, n.5, 1987, pp.356-365

[5] Rapporto ASSINFORM sulla situazione dell'informatica in Italia, 1987

[6] C.A.R. Hoare, An axiomatic basis for computer programming, Comm. ACM, 12, 576-583, 1969

[7] J.W. Lloyd, Foundations of logic programming, Springer, 1987

[8] F. Kroger, Temporal logic of programs, EATCS Monographs, Springer, 1987

[9] M.R. Garey, D.S. Johnson, Computers and Intractability, W.H. Freeman and Co., S. Francisco, 1979

[10] S. Goldwasser, S. Micali, C. Rackoff, The knowledge complexity of interactive proof systems, Proc. ACM Symp. on Theory of Computing, 1985

[11] A.N. Kolmogorov, Three approaches to the concept of "The Amount of Information", Probl. of Inf. Transm. 1/1, 1965

[12] P. Erdös, J. Spencer, Probabilistic Methods in Combinatorics, Academic Press, New York,1974

[13] G. Brassard, Modern Cryptology, Lecture Notes in Computer Science 325, Springer, 1988