di Raffaello De Masi

Elementi di Prolog

Una decina di anni or sono, nell'oscurità di un garage, due giovani di bell'aspetto e di grandi speranze mettevano insieme una «mela»; si trattava di un esserino piccolo, debole, collegato ad un cigolante registratore a cassette, senza gran memoria, un po' limitato nella espressione (capiva solo se gli si parlava in maiuscolo) e con la strana abitudine, propria delle persone senza grande istruzione, di scrivere in formato manifesto. Era il primo atto dell'introduzione sul mercato del personal computer, piccolo allora anche nella sostanza oltre che nella forma, che all'inizio fu capace di prestazioni limitate, parlava solo un Basic un po' deboluccio e disponeva di memorie di massa (si fa per dire) formato blocco-note. Chi comprava la macchina doveva dotarsi di pazienza cistercense e scriversi i suoi programmi (al massimo il mercato offriva un word processor), o pregare di farlo qualche suo amico praticone, che, sudando le classiche quattro camice (più altre quattro di ricambio) alla tastiera, tirava fuori un codice grossolano, illeggibile (anche per chi l'aveva scritto, dopo qualche settimana), che, bene o male, faceva quello che si desiderava facesse senza troppo uscire dal seminato

Sono passati dieci anni, ed oggi un mega di memoria centrale è il minimo che una macchina deve possedere per farsi vedere per strada, si è divenuti raffinati nel dire e nell'esporre, la multiprogrammazione ed il multitasking sono sul ballatoio di casa che aspettano, un linguaggio, per essere serio, deve essere compilo-interpretato, gli pseudoprogrammatori della domenica sono spariti per fare spazio alla più avanzata professionalità, e se proprio è necessario scriversi un programma, è inutile imparare o scegliere un linguaggio; basta comprare un generatore di programmi che farà, nella maniera più giusta, pulita ed elegante il lavoro per noi.

Che bello; d'altro canto è questa la legge di tutte le nuove invenzioni! Agli inizi del secolo il guidatore gentleman (così si chiamava, e tale era, al contrario di tanti automobilisti che ci tocca incontrare oggi per strada), era, come l'utente di un computer di dieci anni fa, capace di mettere le mani nella sua macchina, da eccellente meccanico; oggi mia moglie è stata capace di mettere benzina nella sua Mini Diesel («Tanto, non è più o meno la stessa cosa»?) e, probabilmente, ha solo una vaga idea che, ogni tanto, occorre sostituire l'olio o controllare il liquido di raffreddamento. D'altro canto in linea di massima, anche lei non ha tutti i torti; man mano che si va avanti nell'evoluzione di una macchina, occorre essere sempre più specialisti, oppure lasciar perdere e affidarsi ad altri. Nessuna meraviglia, oggi, se si parla quindi, nel campo degli idiomi per calcolatori, di linguaggi specialistici; ed è giusto che ci siano, come è giusto che chi intende discutere di filosofia impari innanzitutto i termini di questa disciplina.

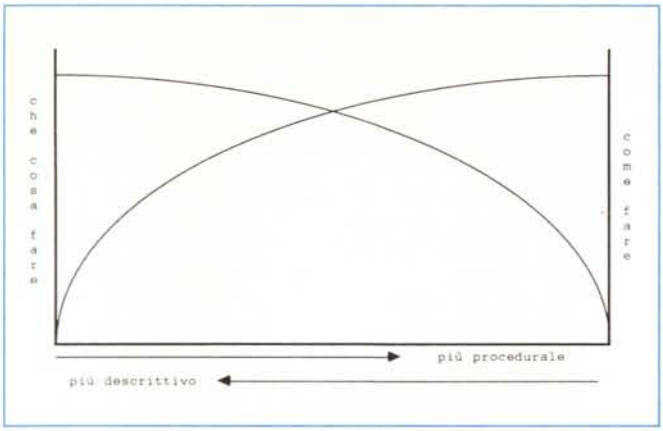
«Eccone qui un altro», dirà qualche lettore, «che dopo la sviolinata iniziale ci viene a dire che il Prolog è il miglior linguaggio perché bla, bla, bla». Vero, e falso; vero in quanto dirò che davvero Prolog è il miglior linguaggio (ma aggiungendo per certe applicazioni), falso perché non mi interessa in particolare dimostrare che Prolog è più facile o difficile di un altro; Prolog è solo «differente», diverso nella forma e nella sostanza da qualsiasi linguaggio apparso sul mercato, ivi compreso quel Forth (lo ricordate) di cui parlai su queste stesse pagine qualche anno fa, o quel buon «C» dell'ancor più buono papà Corrado.

Ho cominciato ad interessarmi di Prolog un paio d'anni or sono, studiando certi algoritmi di intelligenza artificiale, e mi parve subito ben più interessante e pratico del LISP, universalmente adottato in questo campo di problematiche. Interessante in quanto, a differenza di questo, è molto più intuitivo, pratico in quanto è, ancora di questo, molto più elastico e disponibile a diverse applicazioni (tanto per intenderci, si trova molto più a suo agio nel campo matematico, non proprio adatto alle aree ed alle possibilità del LISP). È, del LISP, inoltre, molto più moderno, anche se non gode di altrettanta fortuna in campo universitario; e, infine, il che non guasta, molto più standardizzato; in breve, si presenta tanto interessante da meritare una sbirciatina ben più da vicino; è quello che contiamo di fare in queste puntate, spalla a spalla col «C» di Corrado.

Un po' di storia

La mente umana è un organo eccezionale, meraviglioso computer della inimmaginabile (è il caso di dire) potenza, ciononostante portatile (pesa circa un chilo e mezzo), funzionante sempre in tempo reale, dalla memoria pressoché illimitata, anche se non sempre funzionante al meglio, fornito già di un contenitore a tenuta antiurto, atermico, climatizzato, e dotato di periferiche e di sistemi di I/O raffinatissimi e di ragguardevole potenza. L'alimentazione è autofornita dal suo supporto, e un gruppo di continuità a pompa lo rifornisce ininterrottamente di energia per periodi anche superiori, talora, al secolo.

Niente male come realizzazione, visto che si tratta di una macchina, oltre tutto, che lavora in vero multitasking, con un numero di job concorrenti praticamente illimitato.



Tipo di programmazione in programmi procedurali e non procedurali.

Lo standard di questa macchina non è fisso e, specie nell'area della programmazione autonoma, i risultati non sono sempre eguali per tutti; anche i contenitori in cui è fornita sono sempre diversi l'uno dall'altro (guai se non fosse così!), ma l'aspetto esterno di essi non ha alcuna correlazione con la potenza della macchina stessa. Circa la metà dei modelli presenti sul mercato pesano un centinaio di grammi in meno, grammi che, a detta dei modelli più pesanti, si sentono mancanti tutti. Ciononostante si tratta di strutture di incredibile complessità, e, proprio per questo, purtroppo non sempre facili da riparare.

David Ritchie, nella sua eccellente opera «The Binary Brain; Artificial Intelligence in the age of electronics» effettua una disamina molto accurata del fenomeno mente umana, della capacità di questa di sottomettere il creato. Si tratta di una piacevole lettura che, alla fine, giunge alla conclusione che l'uomo e la sua mente hanno raggiunto l'assoluta supremazia grazie a due cose; la padronanza del fuoco e la capacità di costruire oggetti e tecnologia sempre più complicata ed efficiente.

L'uomo si è trovato, alla fine, a scoprire i limiti del pur potente computer di cui il Padreterno l'aveva dotato; o, per meglio dire ne ha scoperto uno, forse il più banale; ha pensato allora di crearsi un suo aiuto, un sosia, quasi, più potente dove lui era incapace, secondo un principio che ha d'altro canto animato e determinato la creazione di tutti i tool di cui aveva bisogno; migliorare l'efficienza di alcune sue non proprio eccezionali prerogative.

Ha costruito così il computer, gli ha affidato da molto tempo i suoi problemi di bruto e veloce calcolo e senza perdere tempo, ha affrontato un gradino superiore della conoscenza, la realizzazione di una macchina pensante (pur in certi limiti) autodeterminantesi. Il progetto era ambizioso, pur disponendo di macchine sofisticatissime quali quelle esistenti oggi; ed i risultati non sono stati immediatamente risolutivi ed esaustivi. Ma non si può pretendere di più, se si considera la complessità del problema.

Ma, come in ogni progetto di realizzazione di qualcosa, anche in questo, l'uomo aveva bisogno di tool; si è costruito così dei linguaggi, mezzo principe per conversare con la macchina, attraverso cui instradare questa nella soluzione dei problemi che a lui interessavano. PLANNER, IPL, INTERLISP, SAIL, KRL, sono solo tessere di un mosaico di idiomi destinati a realizzare nella maniera più acconcia e conforme questo nuovo connubio intelligente uomo-macchina. Quale scegliere?

L'intelligenza artificiale e di riflesso i suoi linguaggi, sono rimasti per molto tempo chiusi nelle aule universitarie; ma l'inflessibile legge della richiesta commerciale ha funzionato come il miglior crivello, facendo uscire, più o meno recentemente, da queste, il LISP ed il PROLOG, che oggi, sono disponibili anche su microcomputer della classe MS-DOS et similia. Altri linguaggi sono caduti nel dimenticatoio, ma il principio e lo scopo sono rimasti sempre gli stessi: la messa a punto del miglior linguaggio destinato a problematiche di intelligenza artificiale.

Intendiamoci bene; è possibile redigere programmi «intelligenti» in qualsiasi linguaggio (così come è possibile fare didattica in Basic e Fortran, od applicazioni in tempo reale magari in Cobol ed Algol, o utilizzazioni commerciali in «C» o Forth), e sovente, fino a qualche tempo fa, ben noti programmoni di intelligenza artificiale circolanti nelle aule universitarie nascondevano (come ebbe modo di farmi acutamente notare il mio buon amico Stefano Cinti) certi listatoni in Pascal e, magari, anche in Basic, da terrorizzare anche il buon Spielberg. Fatto sta che, è inutile nasconderlo, padroneggiare completamente il Lisp non è facile, ed il Prolog (ante Borland) era una cenerentola mal vista, anche grazie ad alcuni linguaggi non proprio ben funzionanti. Borland, solo l'anno scorso, ha dato uno scossone a questo mondo un po' cristallizzato e se vogliamo, bigotto, addirittura «dissacrando» il Prolog e fornendone una «vulgata» all'utente MS-DOS. Il bello sta nel fatto che il Turbo Prolog, questo enfant terrible dei linguaggi, lungi dall'essere intimorito dai suoi sussiegosi omonimi togati, si è fatto avanti in gran forma, oscurando i suoi predecessori con una notevole messe di potenzialità e con prestazioni e facilità d'uso certo non presenti altrove. Se questo, credo, ha dato fastidio a qualche testa coronata della «intelligentia» informatica, che si è visto invaso da turbe di utenti il proprio campicello dove credeva di poter dettar legge con la sua scienza, ha però dimostrato che non esiste nulla di tanto complesso da non poter essere descritto nella maniera più semplice ed accattivante. Questi appunti, che presenteremo nel corso di qualche puntata, vogliamo sperare possano servire, ad un utente ancora all'oscuro del mondo del Prolog, a fargli gustare una boccata d'aria diversa (proprio perché ci sentiamo di affermare, senza tema di smentite, che chi entra in questo nuovo mondo imparerà a respirare in maniera davvero diversa).

Note tecniche d'inizio

Queste puntate parleranno di Prolog tout-court, inteso come linguaggio specifico ed avulso da qualsiasi dialetto; purtroppo affermazioni di tal fatta sono destinate a rapida smentita, visto che è gioco forza, anche nei linguaggi più standardizzati, far riferimento ad una particolare implementazione; perciò ridimensioneremo il nostro dire, affermando che, accanto ad una trattazione il più generica possibile, faremo riferimento comunque al Turbo Prolog della Borland; il motivo è presto detto: questo linguaggio sta divenendo, a ragione, lo standard de facto del linguaggio, visto che le non numerose implementazioni presenti sul mercato, e principalmente destinate ai grossi mainframe universitari, da una parte sono troppo poco note, nelle loro caratteristiche, per rappresentare uno standard universale, dall'altra, paradossalmente, il Turbo possiede potenzialità e caratteristiche avanzate, tanto da coprire le migliori caratteristiche delle diverse implementazioni.

Del Turbo (che strana, questa denominazione) abbiamo già parlato a lungo nel numero di dicembre 1987, quando ne eseguimmo la prova su queste pagine. Il pacchetto, nella sua primitiva edizione (che è tuttora quella corrente, la 1, nella revisione [.1]) è rappresentata da due dischetti, contenenti l'uno il linguaggio vero e proprio, l'altro una libreria di programmi, e da un nutrito manuale, redatto nella classica grafica della Borland e disponibile in versione italiana (circa 300 pagine), nel qual caso il pac-

chetto è corredato di tre dischi. Ne esiste la sola versione dedicata al mondo MS-DOS, mentre, al contrario di quanto avviene per altri pacchetti della stessa Borland, non pare prossima l'implementazione su Macintosh. Come tutte le implementazioni Borland, il software non è protetto e ciò consente una agevole installazione su hard disk, cosa consigliabile, appena possibile, specie per esplorare l'ampia messe di programmi e demo di libreria, forniti con il pacchetto.

Il Prolog

Sebbene l'età pionieristica dell'acquirente-analista-programmatore tuttofare sia (fortunatamente) tramontata da un pezzo, si nota, mai come oggi sul mercato, una fioritura estesa ed articolata di linguaggi di programmazione. Il motivo di questo proliferare è sottile e non proprio ovvio, e soggiace alla domanda: Come mai, in un'epoca in cui si trova tutto già pronto sul mercato Borland riesce a vendere ancora un milione di copie del suo turbo Pascal?

I motivi sono molteplici, e diversamente articolati; come acutamente fa notare un manuale di un linguaggio, nel 1970 il software rappresentava circa il 10% del costo totale di un sistema, mentre oggi il prezzo dell'hardware è caduto precipitosamente ed il software può rappresentare anche l'80% del capitale totale da investire per informatizzare una qualsiasi attività. Questo rapido rifiorire dell'attività professionale di programmazione e questa lievitazione di costi ha stimolato lo sviluppo di nuovi mezzi di programmazione atti a rendere più facile, veloce, e, di conseguenza, competitivo, redigere un programma (non più di dieci anni or sono la programmazione professionale doveva essere svolta nel durissimo Assembler, per possedere crismi accettabili di velocità ed efficienza; oggi chi lo farebbe più avendo a disposizione, tanto per fare un esempio, il «C»?); tanto per intenderci, gli stessi nuovi (e non tanto nuovi, vedendo l'esempio di UNIX) sistemi operativi vengono progettati e redatti in linguaggio evoluto.

Nello sviluppo di sistemi esperti e di intelligenza artificiale, oggi Prolog è divenuto insostituibile; esso è il risultato (come avemmo già modo di narrare) di una lunga serie di ricerche guidate da un ricercatore dell'università di Marsiglia, Alain Coulmerauer, che, agli inizi degli anni settanta, affrontò il problema della realizzazione di un tool per la PROgrammazione LOGica (da cui il nome). I risultati furono subito eccellenti se si considera che molti shell di sistema esperto, come APES, ESP/Advisor, KI, BPr, ed altri sono stati redatti direttamente in Prolog. Si tratta di un linguaggio di gran lunga più efficiente e potente della maggior parte di quelli esistenti, tra cui anche il Pascal, Basic, e lo stesso «C». Per utilizzare un esempio citato dalla stessa Borland, un programma in Prolog richiede, pur conservandone la stessa chiarezza, un numero di righe almeno dieci volte minore rispetto a quelle necessarie in Pascal.

La validità di questo linguaggio è confermata ancora dalla sua scelta come idioma di base, da parte dei giapponesi, per la progettazione e la produzione di calcolatori della quinta generazione, nei quali funge addirittura come linguaggio di sistema (dove praticamente tutti quelli odierni, non UNIX, adottano invece, l'Assembler). Comunque, a differenza di quanto avviene con altri linguaggi, dove la conoscenza di tecniche, anche solo di base, di programmazione in altri idiomi è utile, imparare il Prolog è più semplice, paradossalmente, per coloro che si avvicinano per la prima volta alla programmazione che per coloro che già conoscono, autodidatti, un altro linguaggio evoluto, tipicamente Basic, Fortran o Pascal. Il motivo sta nella tipologia d'approccio al problema, che, nel nostro, è del tutto diversa da quanto avviene negli altri.

In Basic, Pascal, C, Fortran, Forth, e in tutti gli altri linguaggi tradizionali un programma è rappresentato dall'esposizione ordinata e più o meno articolata delle regole e delle quantità da manipolare per giungere a 1 risultato desiderato. In pratica, ed in altre parole, si «insegna» al calcolatore il procedimento necessario a conseguire i risultati, obbligandolo a ripetere le sequenze che l'uomo percorrerebbe per giungere agli scopi prefissi. Questo sistema di risoluzione, che impone alla macchina di *procedere* secondo certe direttive, è chiamato appunto «procedurale». Prolog è un linguaggio dichiarativo, vale a dire che tramite esso si descrivono alla macchina le ipotesi del problema e la tesi che si desidera raggiungere, eventualmente aggiungendo i dati; è poi la macchina stessa che realizza una procedura capace di giungere al risultato desiderato, ovviamente raggiungendolo più o meno precisamente in base alla esattezza ed esaustività delle premesse stesse. In altri termini, un programmatore in Prolog impiega più tempo a descrivere che cosa desidera che la macchina faccia che in che modo questa cosa vada fatta, il tutto, il che non guasta, senza troppo tener conto di ordini precisi e successioni pedanti. Prolog affronta i problemi in maniera forse non proprio ordinata, ma certo molto più vicina alla mente umana.

Tanto per sdrammatizzare la cosa, immaginiamo di leggere un articolo in cui si descrive un incidente per lo scoppio di un pneumatico di un'auto. Basic e Pascal analizzerebbero solo l'accaduto traendo le conseguenze che la macchina aveva bisogno di nuove gomme o di una maggiore manutenzione di quelle esistenti; Prolog ci fa venire in mente che anche le nostre non sono in buone condizioni e, inoltre, *che la nostra stessa auto ha anche problemi di carburazione e di frenatura*. Prolog, cioè, come abbiamo già detto, può, autonomamente, trarre conclusioni anche lontane da quelle di partenza e, comunque, non previste dall'utente.

Un approccio relazionale ad un problema può essere esemplificato col seguente ragionamento: si abbia la parola «rosso»: questa parola, pur con un solo significato, può essere associata da una serie pressoché infinita di fatti, relazioni, oggetti, condizioni; pomodori, facce di persone imbarazzate, Ferrari, carote, mele, estratto conto della banca, biglietti da 50.000. Questa è, ovviamente ridotta, la rappresentazione della classe generale degli oggetti, attraverso cui procedere per una ricerca più accurata. Ma se forniamo, adesso, al nostro ragionamento, una ulteriore informazione circa un tipo di olio lubrificante, istantaneamente tutte le informazioni esistenti, tranne una, scompaiono, e l'oggetto «Ferrari» risulta univocamente determinato, e questo non perché abbiamo fornito alla macchina *regole* per il riconoscimento delle automobili di Maranello, ma perché è stato lo stesso linguaggio, attraverso regole di ragionamento che gli avremo precedentemente fornito, a determinare l'associazione finale rosso-Ferrari, non solo (come accadrebbe in un normale linguaggio), ma determinando un ulteriore avanzamento nella procedura deduttiva, con restituzione e messa a disposizione di ulteriori dati in possesso della macchina ed univocamente riferibili all'accoppiata, come ad esempio «cavallino rampante», «Alboreto», «Agip» e così via.

Si intenda, non è detto che una operazione del genere non sia effettuabile utilizzando un linguaggio diverso; la gran differenza sta nel fatto che in Prolog le connessioni vengono eseguite autonomamente, mentre in tutti gli altri casi il linguaggio non ha capacità autonome di deduzione e gli ulteriori abbinamenti vanno regolati da una serie di regole propositive condizionali, del tipo «if aaaaaaa then nnnnnn». È ovvio che questa intrinseca possibilità di un linguaggio di instaurare autonomamente relazioni consente alla macchina di operare in maniera più veloce, ed ai programmi di essere più brevi e compatti.

Con ciò credo sia il caso di fermarci, per questo primo approccio col Prolog. La prossima volta cominceremo a conoscere più da vicino le caratteristiche di base del linguaggio; a risentirci! MC

power & compatibility

PERSONAL WORK STATION 16 e 32 BIT

PX-30
Cpu 8088 10MHz, 256-640K ram, floppy disk 3,5 pollici, hard disk 20-40MB

PX-50
Cpu 80286 8MHz, 512K-1MB ram, floppy disk 3,5 pollici, hard disk 20-40MB

PX-80
Cpu 32 bit 80386 16MHz, 2MB ram, floppy disk 3,5 pollici, hard disk 20-40MB

SUPERMICRO 16 e 32 BIT

IMMAGINERIA

AX-60
Cpu 16 bit 80286 12MHz, 512K-2MB ram, floppy disk 5,25 e 3,5 pollici, hard disk 40-230MB

AX-80
Cpu 32 bit 80386 16MHz, 2MB ram, floppy disk 5,25 e 3,5 pollici, hard disk 40-230MB

UNIDATA

UNIDATA - Via S. Damaso 20 - 00165 Roma

Tel. 06/63.66.58 - Telex 623122 UNITAT I