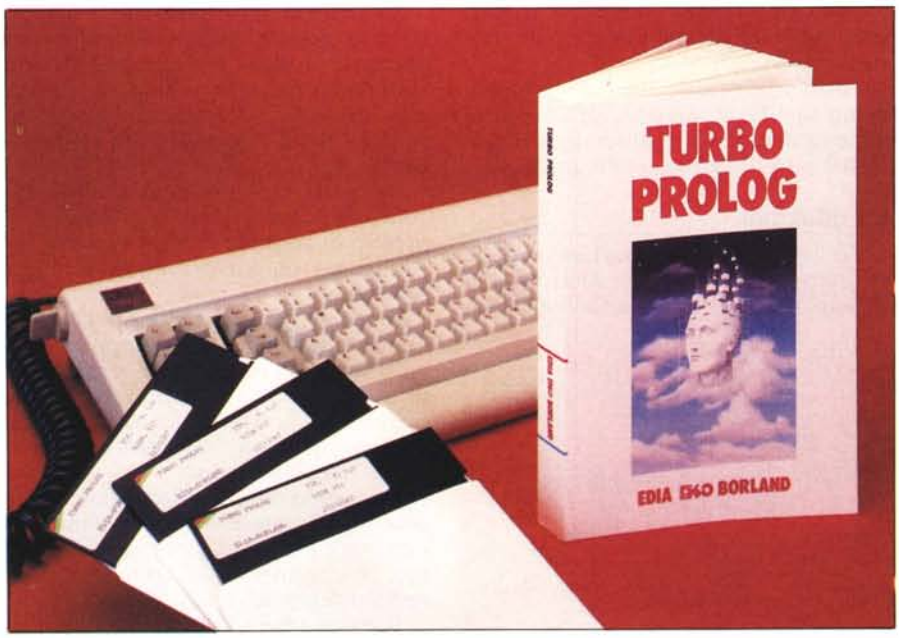
Turbo Prolog



di Raffaello De Masi

Nel 1983 le industrie giapponesi introdussero un ambizioso progetto destinato, secondo le loro intenzioni, a superare il predominio statunitense nel campo dei sistemi operativi e dell'architettura di base delle macchine, avviando una serie di ricerche circa la cosiddetta «quinta generazione» di microcomputer a medio costo.

Linguaggio ufficiale di tale progetto fu il Prolog, primo vero tentativo (tranne il fenomeno a sé del «C») di sostituire il finora padre universale dei sistemi operativi, il pesantissimo Assembler. I risultati sono stati solo parziali, visto che, nel frattempo, l'industria americana non è stata con le mani in mano; ma è importante evidenziare la tendenza dei costruttori a rivolgersi, per i loro bisogni di base di sistema, a linguaggi evoluti, e, nel caso, di livello decisamente molto alto.

Ma la cosa più interessante e curiosa è che, per la realizzazione del progetto, si decise di scegliere non un linguaggio self-tailored (come era accaduto per il «C», che nacque, inizialmente, come mezzo di sviluppo precipuo di un sistema operativo, e solo in un secondo tempo è divenuto linguaggio di programmazione destinato a produrre applicazioni) ma un idioma nato per ben altri scopi, e destinato a campi, come l'intelligenza artificiale, ben diversi da quelli di una gestione di un progetto industriale.

Prima di entrare più propriamente nella illustrazione del pacchetto della Borland vediamo, per sommi capi, che cosa ha determinato la nascita dei linguaggi dedicati alla A.I.

La prima domanda che ci si pone, entrando nel mondo della Intelligenza Artificiale è: «È davvero necessario un linguaggio speciale per questo campo; e, in altri termini, quali sono le esigenze che determinano la necessità di dover disporre di un linguaggio diverso da quelli, già tanto numerosi, presenti nel mondo informatico?»

La necessità di disporre di qualcosa di diverso è dovuta a certe caratteristiche e certe esigenze della A.I. In particolare, per la costruzione di sistemi esperti, espressione più raffinata delle tecniche di A.I., occorre disporre di alcune caratteristiche speciali che possono essere così riassunte:

— sono necessarie nuove varietà di «tipi», per descrivere e manipolare la grande e differenziata messe di dati caratteristiche di questo campo dell'informazione.

— La capacità di dividere, ben più che nelle altre applicazioni, il problema in una serie, per quanto possibile numerosa, di sottoproblemi; l'A.I. per la particolarità dei problemi che è chiamata a trattare, si giova ed abbisogna più di altri campi di linguaggi destinati alla programmazione top-down.

— Strutture di controllo e decisionali potenti (praticamente tutti i sistemi esperti si basano, per la loro quasi totalità operativa, su «scelte»), che attingano alle più avanzate ed efficienti tecniche programmatorie, come la simulazione e la ricursione.

— La necessità di disporre di un linguaggio altamente interattivo; è questa l'esigenza maggiore dei linguaggi di A.I.; già abbiamo visto su queste pagine che il primo vero linguaggio dedicato, il Lisp, è interattivo fino all'assurdo; il Prolog avanza ancora di più su questa strada con tecniche di «analisi interattiva» ancora più efficienti.

— La possibilità di eseguire alcune operazioni particolari, come deduzione automatica o aggiornamento, ancora automatico di conoscenze.

— La capacità di produrre codici compatti, data la tendenza, in questo tipo di problematiche, a «strafare» nelle strutture dichiarative e di analisi.

I linguaggi dedicati presenti sul mercato, le loro connessioni e la loro «genealogia» sono riassunti nella figura A: di questi solo due, il Lisp ed il Prolog, hanno abbandonato il complesso mondo dei mainframe universitari ed il tenebroso labirinto della ricerca, per entrare nell'arena delle piccole macchine personal: il Lisp ed il Prolog; le frecce della stessa figura evidenziano la naturale struttura evolutiva dei due idiomi, ed implicitamente la maggiore o minore potenzialità.

Il Prolog vede la luce nel 1970 all'Università di Marsiglia ad opera di un professore di calcolo numerico, Alain Coulmerauer, tipo alquanto bizzarro per la verità, rappresentando l'archetipo di genio e sregolatezza che non è raro trovare nel mondo informatico. Sono gli anni appena successivi alla grande contestazione giovanile e la ricerca langue soprattutto nelle Università francesi, che sono state la culla di tale grande fenomeno. D'altro canto il Basic (1963) ed il Pascal (due anni dopo) hanno dato una scossa vigorosa al predominio del Fortran e del Cobol, linguaggio quest'ultimo mai effettivamente uscito dalle sue applicazioni commerciali. Con un gruppo di pochi allievi Coulmerauer mette a punto le regole principali del linguaggio in un periodo di 4 mesi, e giunge allo standard completo ed alla sua implementazione finale su una macchina Amdhal nel corso dell'anno. Si tratta di una naturale evoluzione del Lisp, presentato quest'ultimo qualche anno prima (1960, McCarthy) con ben altri scopi (inizialmente il Lisp fu inteso come mezzo per definire notazioni matematiche), basato su un vecchio concetto di manipolazione di liste proprio di un linguaggio a basso livello, destinato alle prime implementazioni di intelligenza artificiale, l'IPL (Information Processing Language), nato nel 1960 ed ormai del tutto dimenticato. Scopo dichiarato del Prolog era quello di abbinare alla manipolazione delle liste la capacità di «deduzione» dai dati che il programma stesso possiede o riceve in Input. Tanto per intenderci la maggior parte dei linguaggi si basa,

per lo sviluppo delle problematiche che sono chiamati a risolvere, sullo sviluppo di regole e sulla manipolazione di dati; il programma in sé non trae alcuna conclusione e non deduce dati da quelli già esistenti motu proprio. In altre parole, un programma in Pascal o Fortran deve prevedere, passo passo, tutte le regole necessarie per la soluzione del problema, e le deve tutte «contenere» fisicamente e sintatticamente, magari sotto forma di librerie, nel suo listato. Al Prolog, invece, si fornisce una descrizione del problema in termine di connessione di fatti e legami tra dati e parti, ed il programma tenta di connettere tra loro questi dati per trarre tutte le possibili conclusioni.

Riprenderemo tra poco l'argomento; ma ci si consenta un esempio forse banale: ricordate i terribili quesiti della Susi su un noto settimanale enigmistico? Implementare questi problemi (Gianni abita in via Carducci, Antonio è tifoso del Torino, Andrea mangia alle 13, Gianni e Vittorio vanno in ufficio insieme, ecc. con buona pace dei bottiglioni di Novalgina usati per la soluzione) in un linguaggio comune è più difficile che risolvere il problema stesso. Il Prolog si presta ottimamente a risolvere questi compiti, per la sua struttura procedurale, capace di trarre conclusioni da certe premesse insite nelle dichiarazioni iniziali.

Questo mondo nascosto, chiuso nelle sale universitarie, è stato aperto, per primo, agli utenti del PC dalla Borland, che ha messo a disposizione il suo Turbo (chi era costui?); nato con la stessa filosofia degli altri linguaggi (facilità d'uso, completezza della documentazione, mancanza di protezione) se ne discosta per il prezzo, un poco (ma solo un poco) più sostenuto rispetto ai concorrenti Pascal, C e Basic. Del tutto ossequiente alle direttive generali del linguaggio ed alla sua sintassi, se ne discosta per la maggiore potenzialità intrinseca, per la velocità del suo compilatore, e per aver portato avanti fino alla massima disponibilità verso l'utente la sua struttura originale «descrittiva».

Sebbene il Prolog sia un linguaggio essenzialmente dedicato, attraverso il Turbo è possibile, grazie alla sua estrema versatilità, eseguire una serie di operazioni efficaci e complesse e giungere a risultati apprezzabili in molteplici altri campi. Attraverso Turbo Prolog è possibile:

— produrre modelli per ogni tipo di applicazione, scientifica, industriale, o di monitoraggio o controllo di macchine;
— produrre database relazionali dinamici;
— procedere all'analisi di linguaggi, umani o di macchine;
— produrre interfacciamenti tra macchine ed utenti più facili e potenti, grazie anche alla capacità di analisi di processi logici insita in tale famiglia di idiomi;
— costruire sistemi esperti;
— costruire pacchetti di manipolazione di simboli e dati;
— produzione di tool di analisi e ricerca in campi sperimentali o di avanzata applicazione.

Siamo molto lontani, vero, dal mondo del Basic e del Pascal? È come aprire la porta di una officina dotata di mezzi strani ma efficienti. È questa l'impressione che abbiamo avuto facendo qualche tentativo in questo linguaggio strano, dalla sintassi inusuale ed esoterica, ma nonostante tutto amichevole e facile da apprendere, proprio perché non pretende assolutamente di far pensare l'utente «a mo' del calcolatore».

Turbo Prolog
rev. 1.1 (5/1987)
Borland International Inc.
4585 Scotts Valley Drive
Scotts Valley
LA 95066

EDIA BORLAND s.r.l.
viale Cirene, 11
20135 Milano
costo L. 249.000 (+ 9% IVA)

Che cosa è un programma in Turbo Prolog

Per poter redigere rapidamente e senza errori un primo (e successivi, più perfezionati) programma in Turbo Prolog occorre tenere a mente una caratteristica importante: questo linguaggio è assolutamente descrittivo: al contrario della maggior parte dei linguaggi evoluti non è immediatamente necessario esprimere un problema in termini di valori da manipolare (siano essi valori numerici o di stringa); occorre invece redigere un programma che consiste in una «descrizione» del problema. Una tipica descrizione si compone di tre parti:

— nomi, parti, o strutture di oggetti utilizzati dal problema;
— relazioni esistenti tra gli oggetti;
— regole che governano queste relazioni.

Ad esempio, indicando al programma i seguenti fatti:

A MC comanda Marco
Giò lavora come segretaria di redazione

introducendo la regola

Lello obbedisce a Giò se Giò scrive una lettera per Marco

lo shell Turbo Prolog deduce

Lello lavora per MC
Lello obbedisce a Marco.
(Pare vero, n.d.r...).

In altre parole, durante l'esecuzione, un programma in Turbo Prolog analizza tutti i dati a sua disposizione per trarne le dovute correlazioni e deduzioni. Inoltre, al contrario di tutti gli altri idiomi, un programma in Turbo Prolog è un programma autoistruentesi, in quanto una volta giunto ad una possibile conclusione, torna indietro per vedere se, avendo a disposizione tale deduzione può intraprendere nuove strade analitiche e giungere a nuove soluzioni. Un ulteriore vantaggio è quello di essere un linguaggio dotato di una struttura e di una sintassi ridottissima, basata sul linguaggio naturale umano e ad esso rifacentesi anche nell'ampliamento del patrimonio di libreria di base.

Turbo Prolog è un linguaggio compilato; ciononostante possiede, come tutti i Turbo, una particolare struttura editor-compilatore che consente (forse qui un po' meno elasticamente che in altre implementazioni) di fare a meno della via crucis editor-linker-compiler e viceversa, in caso d'errore. A questo gran vantaggio si aggiunge il fatto di avere a disposizione un editor di rara efficienza e facilità d'uso, clone di quello già visto nel Pascal, che rende il calvario del debug agevole da sopportare.

Al lancio, Turbo Prolog mostra una finestra in cui, in default, vengono settati i parametri standard IBM; apposite opzioni di configurazione permettono di variare parametri ed opzioni (scherzosamente il manuale si scusa, tra le righe, se i colori scelti per il default, già visibili in login, non sono di gradimento dell'utente, ma anche a questo è possibile porre rimedio). Siamo in menu principale (main) cui è possibile sempre ritornare mediante il tasto di «escape», e che, secondo la tradizione Borland, è dotato di una serie di comandi richiamabili tramite la prima lettera (evidenziata, tra l'altro, in colore diverso), o tramite i tasti cursore (Turbo Prolog non supporta, come standard, almeno in questa release, l'uso del mouse). Le finestre sono, in default, giustapposte, ma sono tutte ridimensionabili singolarmente e sovrapponibili, con prevalenza della finestra attiva sulle altre. L'editor, come in tutti gli altri linguaggi Borland, è molto simile ad una versione semplificata di WordStar (o Multimate). Le altre finestre rappresentano l'area di esecuzione interattiva del programma (Dialog), l'area dei messaggi (di errore, ad esempio, o di visualizzazione delle fasi di sviluppo delle operazioni interne; ad esempio, durante la compilazione vengono evidenziate le fasi di caricamento di moduli esterni, di linking ecc.), ed ancora l'area di Trace

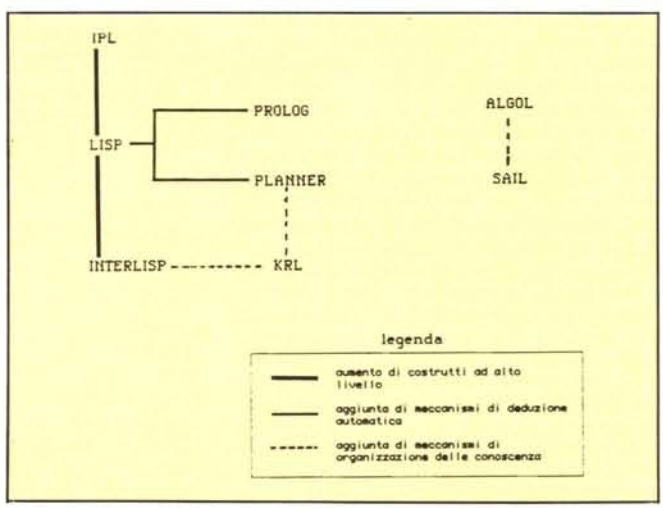
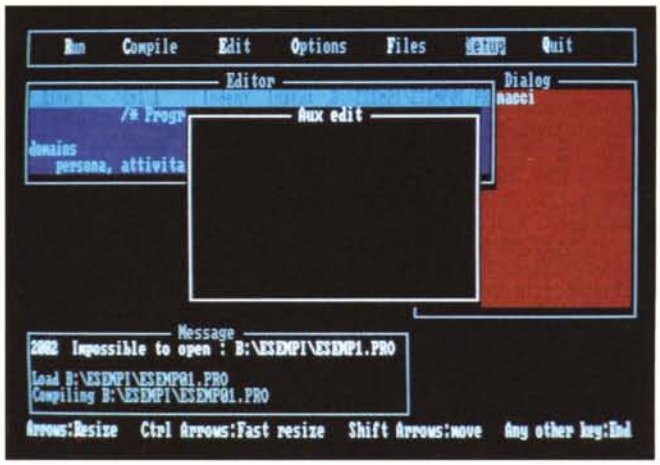
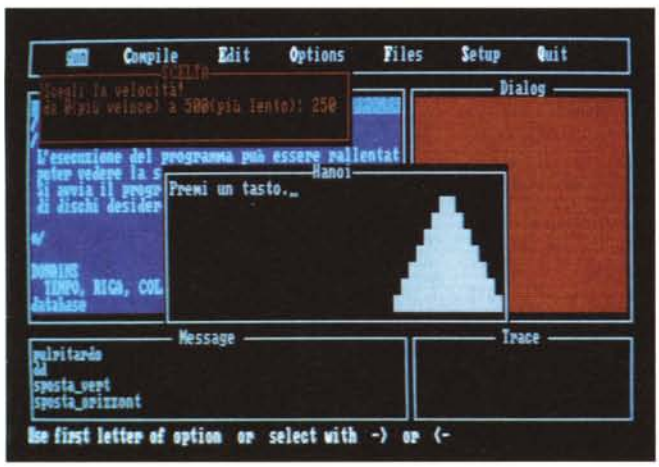


Figura A Interconnessione ed evoluzione dei linguaggi di A.I.; il diagramma va inteso, come sviluppo, dall'alto in basso e da sinistra a destra. Da E RICH, Artificial Intelligence, MGraw-Hill.





Caricamento e lancio di un programma, con sviluppo delle fasi evolutive nelle relative finestre. Si noti l'attività di Message.

Ridimensionamento delle window, per mezzo dell'opzione Setup.

Esecuzione (RUN) di un programma particolarmente complesso come la Torre di Hanoi; è possibile regolare la velocità di sviluppo dei passaggi da 1 a 500; in questo caso il ciclo, con una torre di 7 dischi, si è concluso in 15 secondi. ◀

(inattiva fin quando non richiesta con la apposita opzione). Infine la riga di fondo è riservata ai messaggi di runtime, e, in funzione della localizzazione del cursore, fornisce il messaggio o il consiglio più immediato. Sotto questo punto di vista il linguaggio dispone di un Help in linea abbastanza efficace.

Il manuale d'uso

Borland ci ha abituato fin troppo bene con i suoi manuali: occorre dirlo se si considera che quasi sempre, acquistando un linguaggio, si trova, immancabilmente, alle prime pagine, la fatidica frase: «Questo manuale non è un tutorial del linguaggio, per cui è necessario acquistare uno dei libri in commercio, ecc. ecc.», e, sovente, manca addirittura qualunque referenza o bibliografia. Ciò accade sempre più spesso anche per i linguaggi più semplici e facili, come Basic e Logo. Sembra che i costruttori intendano il manuale come un di più, offerto magari in omaggio all'acquirente, cui non è necessario dedicare più di tanta attenzione.

Borland ha adottato, per tutti i suoi linguaggi (si vedano, addirittura, per il Pascal le preziose guide di riferimento ed introduttive, a corredo del linguaggio principale) il principio di rendere autosufficiente il programmatore che acquista il pacchetto, anche se è alle prime armi. Prolog non manca all'appuntamento, con un manuale che è insieme guida di riferimento e tutorial; sono dedicati alla iniziazione a linguaggio ed al suo pieno sviluppo ben 6 capitoli, una vera miniera di informazioni, molto efficace e corredata di esercizi ed esempi graduati e semplici da apprendere. Non siamo ancora alla perfezione di documentazione del Turbo Pascal (che, oltre tutto, gode di un look, anche tipografico, molto più pregiato), e forse, talvolta, il significato di qualche termine viene dato per scontato, ma uno studio non disattento del volume di circa 300 pagine ci porta a verificare in pieno la complessità (inutile nasconderlo) e la potenza del linguaggio.

La struttura di un programma in Turbo Prolog

Un programma generico in questo linguaggio si presenta sotto la forma:

```
domains
.....
.....
.....
predicates
_ _ _ _ _
_ _ _ _ _
_ _ _ _ _
clauses
= = = = =
= = = = =
= = = = =
```

Queste sono le parti principali, irrinunciabili, di un programma, e, anche in senso lato, corrispondono rispettivamente alle dichiarazioni, alle regole, ed alle relazioni di un programma scritto in un altro idioma, come, ad esempio, il Pascal. A queste parti principali è possibile aggiungere costrutti, strutture, e volendo, il «goal», rappresentato dallo scopo che il programma intende perseguire.

Ad esempio le relazioni Marco - Giò - Lello precedenti possono essere in maniera semplice sintetizzate nel seguente schema di programma:

```
domains
collaboratore, marco = symbol
predicates
lavorare (collaboratore, marco)
clauses
lavorare (corrado, mc)
lavorare (gianni, fiat)
lavorare (andrea, mc)
lavorare (bettino, psi)
lavorare (marco, sua_moglie)
lavorare (lello,X) if lavorare (corrado,X)
```

si noti l'ultimo rigo, che introduce una condizione, d'altro canto facilmente intuibile. Scriviamo il programma alla tastiera e, tramite le frecce cursore, lanciamo il programma stesso. Il programma viene compilato (un altro vantaggio è che il tutto avviene sempre in presenza dell'ambiente editor, per cui eventuali errori e l'inevitabile debug può essere seguito in maniera interattiva) e, nell'area messaggi, compare il messaggio

```
Goal: _
```

Il sistema ci chiede di inserire la nostra richiesta (non è presente, nel listato alcun «goal»). Battiamo:

```
lavorare (lello, mc)
```

Avremo come risposta

```
True
Goal: _
```

Il programma cioè ha tentato di eseguire una analisi della riga contenente l'if; poiché Lello e Corrado sono legati dalla stessa relazione di «lavorare» ecco che, alla richiesta «lello lavora per mc?») il sistema risponde di sì.

Proviamo adesso a battere

```
lavorare (lello, fiat)
```

ed avremo come risposta

```
No Solution
Goal: _
```

Semplice, no, anche se detto in poche righe? Il manuale lo fa in tre pagine; questo

La versione 1.1 del Turbo Prolog

Approfittando della traduzione in italiano del manuale, la Edia Borland, rappresentante in Italia della Borland International, ha inserito, nel volume stesso, le avvertenze e le istruzioni relative alla nuova versione, che nella edizione in lingua inglese erano incluse in un file del dischetto, di tipo README. Le nuove feature, sebbene non numerose, sono però molto importanti, in quanto agiscono in maniera essenziale sulla gestione, tra l'altro, del linker e del compilatore, oltre ad aggiungere una piccola serie di istruzioni di discreta utilità ed interesse.

La versione 1.1 è innanzitutto più veloce, in fase di compilazione, della versione precedente (da circa il 30% fino addirittura, a detta del produttore, al 400%). Inoltre produce un codice leggermente più compatto. Il linker è divenuto invisibile, visto che è possibile scegliere, dal menu Option, l'opzione EXE, per pervenire al codice finale direttamente, senza altro intervento da parte dell'operatore.

In caso di progetti multiprogramma è possibile seguire una compilazione automatica di tutti i moduli, scegliendo l'apposita opzione dal menu a tendina. È possibile ancora usufruire di numerose utility, come rendere deterministico un predicato, commutare istantaneamente il tracing da inattivo ad attivo e viceversa, anche in fase di compilazione iniziata, adottare una configurazione prescelta dall'utente, conservata in un apposito file.

Altre istruzioni consentono di definire posizioni della penna, leggere i parametri della riga di comando usati quando si invoca un programma, eseguire la scansione continua della tastiera in modo abbastanza simile all'INKEY$ del Basic, passare, da programma, da finestra a finestra, eseguire uno scroll controllato della finestra attiva, non solo in su e giù, ma anche a destra e sinistra. Piccole cose, che fanno più dolce la vita (del programmatore)!

per dire la cura con cui viene sviluppato il vero e proprio tutorial. Ma siamo solo all'inizio di uno sviluppo di un linguaggio cui spetta solo il titolo di «diverso»: da qui (siamo a pag. 50) fino alla pagina 220 circa la strada è lunga e talvolta non piana, anche senza essere da sesto grado. Non si tratta di complessità vera e propria; ma la vecchia sintassi statement dopo statement del Pascal o del Basic va dimenticata al più presto.

Caratteristiche specifiche del linguaggio

Con l'esposizione del linguaggio nella sua struttura formale dobbiamo fermarci qui. Diremo solo che, in successione, vengono esposte nel manuale le tipologie e le regole che regolano (che brutta ripetizione) i domini, gli oggetti e le liste (si tratta di strutture simili agli array di linguaggi più convenzionali). L'occasione (ed il capitolo) è buona per una analisi approfondita della ricorsione, che si trova avvantaggiata preziosamente dalla presenza delle liste stesse. Dalla fase dichiarativa si passa poi all'analisi speculativa, vale a dire alla ricerca delle soluzioni nei modi consentiti dal linguaggio stesso, anche tramite l'uso diffuso ed avanzato del backtracking (struttura d'analisi regressiva nelle liste). Il sistema, in pratica, durante il tentativo di analisi di una lista (che, lo ricordiamo, può essere composta di altre liste, anche ulteriormente nidificate; è il principio, in fondo degli array), procede a ritroso nella struttura, alla ricerca di elementi che possono soddisfare ai suoi scopi: Turbo Prolog supporta un efficace tool, il cut (che, in codice, si presenta come un punto esclamativo [!]) che impedisce il bt oltre tale punto.

Ancora, file e stringhe, che nel manuale sono presentati e trattati nello stesso capitolo, sono trattasi in maniera estesa e circostanziata. Anche qui la concezione di base è diversa e merita, a scanso di delusioni, una attenta analisi, con abolizione di tutti i pregiudizi provenienti da precedenti esperienze. Il fatto di star parlando di file ci porta a scoprire una caratteristica insospettabile di TurboProlog; è possibile interfacciare procedure scritte in altri linguaggi e presenti, sotto forma di file, su disco. Gli idiomi supportati sono Fortran, C, Assembly e, poteva mancare, Pascal.

Completa il manuale una guida di riferimento molto ricca ed esauriente, oltre ad una guida rapida all'editor ed ad alcune note sullo stile di programmazione e sulle tecniche di debug.

Conclusioni

Turbo Prolog è un tool d'eccezione per diversi motivi: oltre al fatto di essere estremamente potente (un esempio di costruzione di sistema esperto, facile da comprendere e di notevole eleganza ed efficienza viene fornito nel manuale e sul dischetto) e gode di una documentazione di prim'ordine rappresentata, oltre che da un manuale efficiente e ben realizzato, da una notevole mole di esempi, anche sotto forma di programmi già redatti e forniti su due dischetti di accompagno. Il numero esteso di esempi e la loro struttura notevolmente differenziata (si va da una ricerca tra 7 elementi in base alla loro struttura ed alle loro caratteristiche fisiche al controllo completo di un DB «intelligente» altamente efficiente e potente) rende l'apprendimento semplice e graduato, se si ha l'accortezza di dimenticare vecchi schemi propri di linguaggi rigidi; sembrerà strano, ma programmare in Prolog è generalmente più facile per chi proviene dal Basic o dal Fortran che per il cultore del Pascal; sotto questo punto di vista Pro (come viene detto tra pochi intimi) è il linguaggio più libero che si conosca, visto che nessuno, forse, come lui, si avvicina al linguaggio naturale. Questo crea qualche difficoltà per chi è abituato a programmare per regole, visto che invece dovrà imparare a programmare «per relazioni». Ma superata la boa della incomunicabilità si apre lo spazio immenso, è il caso di dire, della libertà del linguaggio.

MC

Il package Turbo Prolog

Oltre al manuale, più volte nominato, per la sua autorevolezza, completezza e chiarezza in queste pagine, la versione 1.1 del package è formata da tre dischetti, contenenti i seguenti file/directory:

PROLOG.EXE, contenente editor, compilatore, gestore dei file per la fase di runtime.

PROLOG.SYS, contenente le opzioni di default o di settaggio dell'utente relative a finestre, colori, ecc.

PROLOG.ERR, file contenente i messaggi d'errore. Sebbene il sistema utilizzi valori dell'ERRN fino ad oltre 5000 (dec), in effetti i codici utilizzati sono circa 200.

PROLOG.HLP, contenente i messaggi di aiuto; redatto in puro ASCII, può essere aperto da un WP e modificato secondo il piacimento dell'utente.

README, come al solito, aggiunge le correzioni o i consigli dell'ultima ora, assenti nel manuale.

PROLOG.OVL, file di overlay necessari nel corso di sviluppo e di compilazione dei programmi.

PROLOG.LIB, contiene la biblioteca necessaria quando si collegano file oggetto (OBJ) per creare file eseguibili (EXE).

INIT.OBJ, rappresenta un file da inserire in tutti i comandi di linking, quando si creano programmi EXE, e contiene una serie di istruzioni destinate ad inizializzare la RAM disponibile prima di elaborare l'effettivo codice EXE di programma applicativo.

ESEMPnn.PRO, disposti in subdirectory, sono numerosi esempi di programmazione, molto ben graduati in difficoltà ed uso delle risorse. In particolare un programma, GEOBASE.PRO rappresenta una efficiente palestra di apprendimento ed una inesauribile miniera di spunti, per la sua complessità e l'esteso uso delle risorse del linguaggio.