di Raffaello De Masi

Il controllo dell'editing sullo schermo

Ci avviamo a rapidi passi verso la conclusione di queste nostre chiacchierate sul Prolog e sul Turbo Prolog. Tempo ne è passato parecchio, da quando all'inizio cominciammo a parlare di goal, di predicati, di funzioni, presenti in questo particolare linguaggio, così diverso e anticonvenzionale. Tempo ne è passato, dicevamo e abbiamo visto le possibilità di questo idioma, che, al di fuori delle pur vaste aree dell'intelligenza artificiale, possiede caratteristiche che lo rendono all-purpose, come d'altro canto sta accadendo per quasi tutti i linguaggi esistenti sul mercato.
Finora però abbiamo sviluppato solo gli aspetti generali del linguaggio, quelli legati a una programmazione all-purpose, appunto, senza approfondire, come ha potuto senz'altro vedere chi ci ha letto, gli aspetti fondamentali dell'intelligenza artificiale.
Parlare di Prolog legato all'intelligenza artificiale sarebbe come parlare di Fortran nell'ottica di risoluzione di formule matematiche; ha e non ha senso contemporaneamente. Non si possono imparare le tecniche dell'intelligenza artificiale imparando un linguaggio dedicato; sarebbe come pensare di saper guidare solo acquistando un'automobile. Anche per questo, poche pagine accanto, c'è una rubrica specifica destinata all'AI con più precisi riferimenti circa le tecniche; Prolog è solo un tool, un mezzo per rendere più facile l'implementazione di queste tecniche. Ma di questo avremo modo di parlare estesamente altrove.
Come avevamo preannunciato la volta scorsa, in queste ultime battute ci interesseremo di controllo dell'editing sullo schermo; si tratta di un argomento di notevole interesse, non foss'altro perché ci facilita molto il lavoro. Vediamo come

L'editing di schermo

Sebbene generalmente non ci sia necessità di accedere direttamente al Dialog Window, Turbo Prolog mette a disposizione due predicati piuttosto potenti, capaci di lavorare e modificare il testo sullo schermo [edit] e [editmsg].

Il predicato [edit] lavora con due valori di stringa. Il primo è una stringa, o una variabile che si riferisce a una stringa. Il secondo è una variabile non inizializzata a un valore, alla quale si desidera inizializzare la stringa quando il predicato ha esaurito il suo compito e le sue funzioni. Chiamando questo predicato, la corrente finestra attiva cambia in un modo abbastanza evidente; la linea superiore della finestra si trasforma a simulare funzioni di editor. Il cursore si sistema sulla prima lettera della stringa; l'utilizzatore può manipolare, battere e riscrivere il messaggio esattamente come se fosse in editor; all'atto pratico, e non è una esagerazione, è come se fosse stata creata una seconda finestra di editor. Dopo aver concluso, l'utente premerà il tasto F10 e la nuova stringa così scritta verrà immagazzinata nella variabile definita nella seconda parte del predicato.

Immaginiamo di battere alla tastiera:

```
Goal: edit(«La lettera A viene prima della lettera Z»,Messaggio).
```

Spostandosi con le frecce cursore dopo la parola «viene» battiamo molto, quindi premiamo il tasto F10. avremo sullo schermo:

```
Messaggio = La lettera A viene molto prima della lettera Z
  True
  1 Solution
  Goal:
```

disporre di tale caratteristica, senza ricorrere all'editor, è di particolare utilità se si considera che un utente della tastiera ben allenato preferirà eseguire l'operazione con grande velocità.

[edit] è il fratello minore (molto minore) dell'altra funzione [editmsg]. Un maggiore livello di complessità viene attinto quando si desidera usare una forma di editing durante l'esecuzione di un programma. Questa seconda funzione consente di usare l'editor in maniera abbastanza simile al predicato [edit], ma aggiunge un help file, un messaggio di prompt, e la possibilità di indicare se l'utente, dopo la modifica, desidera che questa divenga effettivamente definitiva o no. Non a caso questo predicato è uno dei più complessi di tutto il linguaggio. La figura a mostra le caratteristiche e le peculiarità delle parti del predicato.

Sebbene questo predicato non sia stato realizzato per essere usato all'interno di una finestra standard di Turbo Prolog, è possibile immediatamente eseguire una sperimentazione di esso. Nella finestra di editor, pulita, battiamo:

```
Goal: editmsg(«Prova del predicato editmsg»,Stringa1,«Sinistra»,«Destra»,«Vediamo che succede»,0,«file1»,Codicedichiusura).
```

Figura a
Il predicato [editmsg] e i suoi parametri.

Il messaggio «Prova del predicato editmsg» comparirà al top della finestra, e alla base della finestra stessa ci sarà il messaggio «Vediamo che succede». Chiaro?

Spostandosi sul messaggio iniziale con la freccia del cursore il messaggio di fondo scompare. Immaginiamo di cambiare il messaggio stesso in «Verifica del predicato editmsg». Premendo F10 la finestra si modifica interamente nel modo seguente:

```
Stringa 1= Verifica del predicato editmsg,
Codicedichiusura=0.
1 solution
Goal:
```

Il controllo del cursore

Parlare di editing non è solo parlare di schermo e di finestre; è interessante anche parlare di cursore e controllo dello stesso. A tale proposito Turbo Prolog mette a disposizione due predicati, [cursor] e [cursorform]. Il primo permette di riposizionare il cursore nella finestra, o di determinare dove è andato a finire. Il secondo permette, in certi limiti, di ridimensionare il cursore stesso. Vediamo come.

Per trovare dove è andato a finire il cursore nel corso di un programma il predicato [cursor] viene usato con due variabili intere come argomenti. Come prevedibile, la prima variabile conterrà la riga e la seconda la colonna dell'indirizzo del cursore.

Usando invece lo stesso predicato con due costanti intere come argomento il cursore verrà rilocato all'indirizzo determinato dall'intersezione dei due assi riga-colonna.

La forma e la relativa posizione all'interno del rettangolo dedicato a uno spazio è controllato dal predicato [cursorform]. Anche questo predicato, come quello precedente utilizza due argomenti interi che devono essere compresi tra 0 e 7 (se si desidera che il cursore sia visibile). Se si usa un valore superiore, il cursore scompare dallo schermo.

La linea di partenza è il primo valore delle variabili, e la seconda rappresenta la forma del cursore stesso, come numero di lineette rappresentanti lo spessore del cursore stesso. Si tratta, più che altro, di qualcosa relativa a look, e ha ben pochi fini pratici.

Creazione delle finestre

Per la verità, di questo argomento abbiamo già parlato in più riprese. Vediamo di affinare un poco il tiro usando il predicato [makewindow]. Esso richiede otto argomenti tutti tranne uno interi. La figura b mostra la struttura specifica del predicato.

Un esempio d'uso potrebbe essere;

```
goal:
makewindow(1,7,7,«finestra 1»,
5,60,15,15).
makewindow(2,112,7,«finestra 2»,
10,0,8,75).
```

Capire come queste finestre si comportano non è difficile, utilizzando la figura b). Vediamone brevemente le caratteristiche.

Figura b
Il predicato [makewindow] e i suoi parametri.

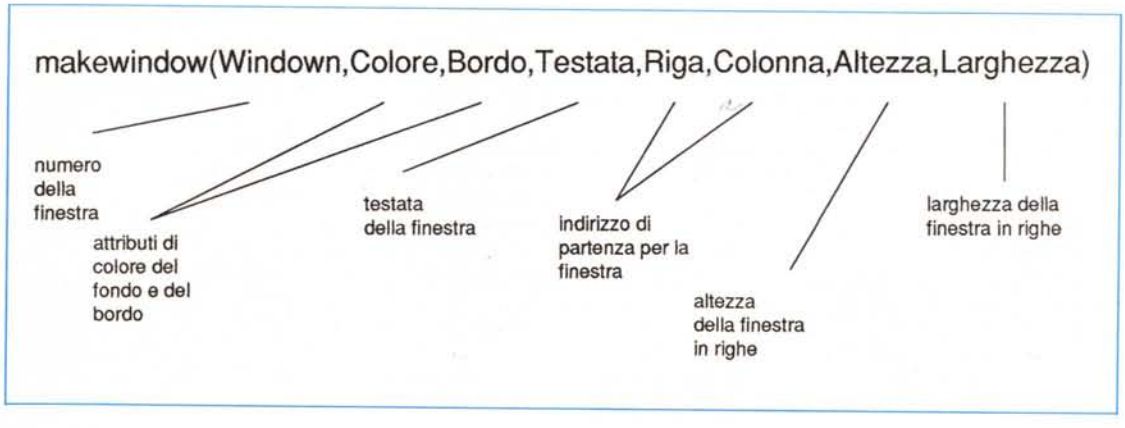

Figura c
Il predicato [graphics] e i suoi parametri.

Numerazione delle finestre: ogni finestra ha bisogno di un numero di identificazione, che deve essere un intero e che deve essere unico nel programma (non ci possono essere due finestre aventi lo stesso numero). Ciò può essere vantaggioso se si organizza il gioco delle finestre con una certa logica, così da assegnare, ad esempio, le finestre da 10 a 20 per operazioni di output, quelle da 20 a 30 per messaggi all'utente, e così via.

Il secondo e il terzo valore determinano il look delle finestre stesse. Il primo determina gli attributi dello schermo (secondo lo schema visto la volta passata), il secondo determina le caratteristiche del bordo, sempre con le stesse regole. Se si utilizza, in questo caso, il valore 0 la finestra non avrà bordo, ma l'assenza di bordo impone poi la mancanza del parametro successivo (che semplicemente non verrà mostrato).

Il quarto valore fornito al predicato (l'unico non rappresentato da un numero) è il titolo della finestra, che viene mostrata in alto al centro della window corrente, come un vero e proprio titolo.

I successivi argomenti del predicato servono a individuare le dimensioni e la posizione della finestra. I primi due determinano la locazione dell'angolo in alto a sinistra della finestra che si sta manipolando; tanto per capirci, [0,0] è lo spigolo in alto a sinistra dello schermo. Gli altri due argomenti specificano invece le dimensioni dello schermo. Il primo mostra l'altezza della finestra in righe, il secondo la larghezza in colonne. Ovviamente, per evitare errori, occorrerà ricordare di non eccedere le dimensioni massime, in particolare 80×25.

Una volta creata la finestra è possibile creare gli attributi in ogni momento con il predicato [window_attr], che consente di settare i parametri appena visti in maniera diversa da quella iniziale. Inoltre una finestra creata nel modo appena visto può essere ripulita del suo contenuto, rimossa completamente, o selezionata e deselezionata attraverso l'uso di tre predicati che vedremo di seguito.

Il primo, [clearwindow] ha ben poco da essere spiegato. Non possiede, ovviamente alcun argomento e la sua funzione è del tutto simile al più generale CLS di DOS. C'è da ricordare solo che alcuni predicati, come [edit] e [editmsg] appena visti, eseguono automaticamente un [clearwindow] prima dell'entrata in azione.

Per cancellare una finestra sullo schermo occorre usare il predicato [removewindow]. Anche questo predicato ha ben poco da essere illustrato. È opportuno ricordare solo di essere ben sicuri che, quando si desidera cancellare una finestra, questa sia quella attiva e non ci siano in essa dati da ricuperare o salvare.

Infine, quando si hanno diverse finestre sullo schermo si desidera passare dall'una all'altra, per renderle attive, è necessario ricorrere al predicato [shiftwindow]. Anche qui è abbastanza intuibile che esso maneggi un solo argomento nella forma.

```
shiftwindow(Variabile)
```

dove [Variabile]; si noti la lettera maiuscola, rappresenta un valore di finestra raggiungibile dal comando stesso.

La grafica in Turbo Prolog

Parlare di grafica in un linguaggio di intelligenza artificiale potrebbe sembrare un controsenso, e in effetti lo è. È ben difficile che, almeno allo stato iniziale, un utente di AI abbia bisogno di grafica sofisticata. In ossequio a queste scarse esigenze la grafica di Turbo Prolog non ha comandi estremamente sofisticati, né d'altro canto lo stesso PC appare particolarmente votato a raffinati lavori di disegno. Approfittiamo, quindi, di questo scorcio di puntata per affrontare e chiudere rapidamente l'argomento.

Per lavorare con la grafica, occorre in Prolog, «entrare» in modo grafico. Ciò avviene utilizzando il predicato [graphics], che manipola tre parametri nel modo mostrato dalla figura c.

Il primo parametro determina il modo grafico in cui si desidera che lo schermo PC lavori. Se si dispone di un monitor CGA è possibile disporre solo dei primi due valori; se si dispone di una scheda EGA si può accedere a tutti e cinque i tipi di risoluzione. Le diverse risoluzioni sono ben note agli utenti del DOS.

Il linguaggio fornisce due predicati principali finalizzati alla grafica; [dot] e [line]; ambedue somigliano molto ai generici comandi di grafica presenti nei più diffusi linguaggi. Il primo manipola il pixel (dipendente dalla risoluzione) attraverso tre argomenti; i primi due sono i valori di riga e colonna, il terzo è il colore al quale viene settato il pixel.

In effetti il predicato [dot] non è molto utile. Molto meglio fa il predicato [line], che permette (in analogia a tanti altri linguaggi) di tracciare una linea retta tra due punti. La sintassi del predicato è molto semplice ed è così rappresentabile:

```
line(PuntoAriga,PuntoAcolonna,PuntoBriga,PuntoBcolonna,Colore).
```

con significato più che ovvio.

Oltre a questi due predicati principali, esiste in Turbo Prolog un package di grafica più potente, che permette di accedere ad un ambiente abbastanza simile a quello in cui si muove la tartaruga del logo. Con una serie di predicati implementati nel linguaggio la «turtle» consente di disegnare in maniera abbastanza intuitiva, «guidando» la tartaruga con ordini (e relativi movimenti) del tipo [left], [forward], [right], [penup] [pendown], e così via. È ancora possibile mescolare grafica e testo, ma il testo non può essere posizionato dove si desidera effettivamente, ma è trattato come un qualsiasi altro testo manipolato dal predicato [write]. Non si tratta quindi, strettamente parlando, di mescolanza di testo e grafica.

Anche stavolta abbiamo completato con l'argomento, che avevamo per la verità affrontato ben due puntate or sono. Ci restano, prima di concludere, da affrontare ancora due argomenti peraltro fondamentali nell'ottica e nell'ambiente del Prolog e, in particolare dei linguaggi di intelligenza artificiale. L'uso dei database e il controllo di flusso del programma (in questo caso per trattare anche del famigerato backtracking); concluderemo infine con le tecniche di debug, e con le direttive del compilatore.

MC

SAMPO KDS 1984

HI-RES 19" COLOR MONITOR

VGA E 8514 COMPATIBILE

MONITOR+SCHEDA"AGA"8514 COMPATIBILE:L.3.900.000

MONITOR+SCHEDA VGA 1024:L.2.736.000

MONITOR AD ALTA RISOLUZIONE 19",RGB ANALOGICO, SI ALLACCIA ALLE FREQUENZE TIPICHE DELLA VGA 640*480 E DELLA VGA ULTRA 1024*768. MEDIANTE SCHEDA 8514 O COMPATIBILE PUO' LAVORARE A 256 COLORI ANCHE NELLA RISOLUZIONE PIU' ELEVATA DI 1024*768 CON I PIU' DIFFUSI PACCHETTI SOFTWARE GRAFICI. LA SCHEDA"AGA",8514/A IBM COMPATIBILE, E' CIRCA 23 VOLTE PIU' VELOCE DI UNA NORMALE SCHEDA VGA E CIRCA 19 VOLTE PIU' VELOCE DI UNA SCHEDA 8514/A IBM.

CRT	Size & deflection	19", 90° deflection angle	
	Display colors	Multicolors	
	Dot pitch	In-line, 0.31mm trio dot pitch	
	Input signals	Analog R.G.B. signals	
INPUT	Data signals	TTL level (positive)	
	Horizontal drive	TTL level (positive/negative)	
	Vertical drive	TTL level (positive/negative)	
Input terminals		15 Pin D Type	
Display area (HxV)		336mm x 252mm	360mm x 270mm 340mm x 255mm (MAC-II)
Display time (HxV)		31.5KHz	400 lines 26.058μs x 13.156ms 350 lines 26.058μs x 11.504ms 480 lines 26.058μs x 15.762ms 35.5KHz 22.80μs x 10.810ms 35KHz 21.164μs x 13.714ms (MAC-II)
Horizontal frequency		31.5KHz/35.5KHz (interlace) 35KHz (MAC-II)	
Vertical frequency		60Hz, 70Hz, 86.96Hz (interlace) 66.7Hz (MAC-II)	
Video amplifier resp.		45MHz (Dot Rate)	
Resolution		720 dots x 480 lines 1024 dots x 768 lines (interlace)	
Display format		640 x 480 Graphics 720 x 400 Text 1024 x 768 (interlace)	
Operator controls		Power on/off, Contrast Brightness	
Power source		AC 120V/60Hz. or 220V/50Hz	
Power consumption		85W	
Dimension (WxDxH)		478mm x 470mm x 441mm	
Weight (net)		24.3Kgs (53.5 lbs)	

EXECUTIVE

PER ULTERIORI INFORMAZIONI CHIAMATE:

EXECUTIVE COMPUTER DEALER
VIA BUOZZI 23 - 22053 LECCO CO
TEL.0341/28.26.14 R.A.
FAX 0341/28.37.59

NEWEL home & personal computer
Via Mac Mahon, 75 - 20155 MILANO
Tel. (02) 33000036/323492 tutto il giorno - 3270226 al mattino - Fax (02) 33000035
Chiuso il lunedì - Aperto il sabato

LISTINO PREZZI TOWER 286 - 386

286 - 1 Mb Ram on board, espandibile a 4Mb, 0 wait state, 16Mhz, 1 drive da 5,25 - 1,2Mb (o 3,5 - 1,44Mb) con controller per FDD e Hard disk. Scheda grafica CGA/Dual Hercules, I/O Plus (RS232 + parallela + clock). Tastiera estesa 102 tasti. Manuali e Dos originale!
Lire 1.450.000

386 sx - (come sopra) 16Mhz	Lire	1.900.000
386 - (come sopra) 20Mhz	Lire	2.350.000
386 - (come sopra) 25Mhz	Lire	2.600.000
386 - (come sopra) 33Mhz	Lire	3.500.000

PARTI AGGIUNTIVE
(differenze in più oltre il prezzo base)

Scheda EGA	Lire	160.000
Scheda VGA	Lire	250.000
Scheda Super VGA (256K)	Lire	300.000
Scheda Super VGA (512K)	Lire	399.000
Hard disk da 20Mb	Lire	390.000
Hard disk da 40Mb (NEC o simili)	Lire	690.000
Hard disk da 40Mb (Quantum)	Lire	990.000
Hard disk da 65Mb (NEC o simili)	Lire	1.190.000
Hard disk da 80Mb (SCSI/ESDI)	Lire	1.360.000
Hard disk da 100Mb (SCSI/ESDI)	Lire	1.900.000
Hard disk da 180Mb (AT-BOS)	Lire	1.990.000
Drive aggiuntivo da 3.5 - 720 Kb	Lire	170.000
Drive aggiuntivo da 3.5 - 1,44 Mb	Lire	230.000

MONITORS

Dual CGA	Lire	199.000
VGA b/n 640 x 480	Lire	350.000
EGA	Lire	700.000
VGA (Normal) 480 x 640	Lire	850.000
VGA (Multiscan) 800 x 600	Lire	950.000
VGA 1024 x 768	Lire	1.100.000
Multisync NEC-2A 800 x 600	Lire	1.200.000
Multisync NEC-3D 1024 x 768	Lire	1.500.000

N.B.
Tutti i Tower sono in Case piccolo.
Case grande + Lire 100.000

I PREZZI SOPRA ELENCATI SI INTENDONO IVA COMPRESA

INOLTRE TUTTI I COMPUTER SONO CORREDATI DI GARANZIA ITALIANA DI 12 MESI

COMPUMAIL®
GRUPPO NEWEL MI

VENDITA PER CORRISPONDENZA SU RETE NAZIONALE - 20020 ARESE (MI) - VIA MATTEOTTI, 21