di Raffaello De Masi

Elementi di Prolog

quinta parte

Lo sviluppo di un programma in Prolog

La volta scorsa abbiamo visto come si costruisce, forse un po' pedestremente, una base di conoscenza, vale a dire le regole formali per «informare» il programma dei fatti, certi e non, di cui può disporre. Mettiamo adesso al lavoro la macchina per consentirle, in base a quanto le abbiamo già dato sotto forma di «fatti», di trarre conclusioni e fornire notizie «ragionate» circa le conoscenze già in suo possesso

Il goal

Già sappiamo che cosa è una base dati; ripetiamo per un momento quella che abbiamo utilizzato, un po' miserella, la volta scorsa, nella quale avevamo inserito anche se in diverse fasi le seguenti parti:

```
Domains
    persone,cose=symbol
Predicates
    mangiare (persone,cose).
    non_mangiare (persone,cose).
    andare (persone,cose).
    bere (persone,cose).
    essere (cose,cose).
    fare (cose,cose).

Clauses
mangiare (andrea,cioccolata).
mangiare (biagio,cioccolata).
mangiare (carlo,cioccolata).
mangiare (alfredo,budino).
mangiare (alberto,meringa).
non_mangiare (ernesto,cioccolata).
non_mangiare (francesco,cioccolata).
non_mangiare (giovanna,cioccolata).
andare (italo,palestra).
andare (luana,palestra).
andare (marcello,palestra).
bere (nora,birra).
bere (olga,birra).
bere (pasquale,birra).
essere_dannosa (birra).
essere_dannosa (cioccolata).
fare_bene (palestra,salute).
```

Che cosa dobbiamo fare per sapere se Olga mangia la cioccolata o se Italo fa_bene? Occorre definire un «goal», un quesito, cui il programma è destinato a rispondere. Le risposte a quesiti come quelli precedenti od altri come «Renzo beve il Last al limone?» possono avere, ovviamente, almeno nel gergo umano, tre risposte: «Sì», «No» o «Non so», quando, ovviamente, non abbiamo conoscenza delle abitudini dei soggetti della nostra domanda. La macchina, di fronte a domande di tal genere, esegue una analisi sistematica di tutte le conoscenze inserite nelle «clausole», vale a dire che analizza tutto il database per vedere se esiste qualche conoscenza in base alla quale mettere insieme una risposta.

Nessuna base di conoscenza, ovviamente, sia quella del nostro piccolo programmino in Prolog, sia quella presente nel più potente ed efficiente computer, può conoscere tutto di tutto. Ciò che una base di conoscenza sa, circa una serie od un solo argomento, è chiamato dominio di conoscenza o, in alcuni casi, «dominio della ricerca».

In base alla tolleranza con cui viene definito un dominio di conoscenza, una base di conoscenza ha la stessa ampiezza (tecnicamente, gli stessi «confini») del dominio; solo alcune volte, per certi usi specializzati, il database è più piccolo. Ogni «goal» deve ricadere nel dominio della base di conoscenza, per poter avere risposta positiva.

La figura B evidenzia la gerarchia di una possibile base di conoscenza nell'ambito delle abitudini ginniche ed alimentari dei nostri amici (esempio che ben si adatta alla piccola base di conoscenza appena definita). La perfetta comprensione di ciò è fondamentale, in quanto, in base a quanto abbiamo detto in precedenza, non è possibile ricavare dal programma deduzioni e notizie di cose che non sono presenti nella base di conoscenza. Tanto per fare un esempio, il nostro piccolo database non possiede alcuna notizia circa altre abitudini di vita dei nostri amici, come, ad esempio, che macchina guidano o che marca di siga-

Figura A

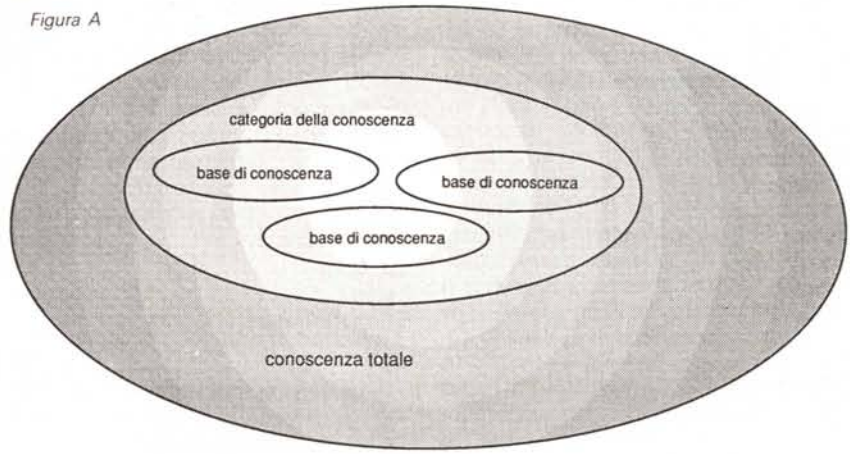
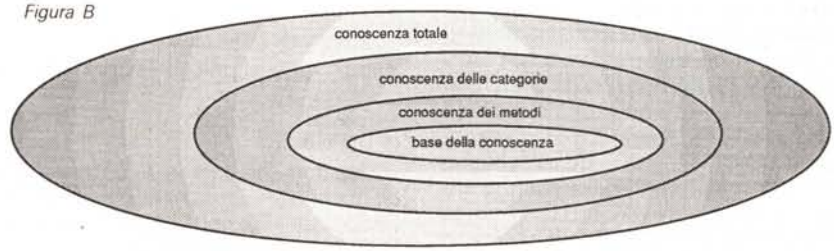


Figura B



Da DAN SHAFER, opera citata.

rette preferiscono. In altre parole si tratta di cose esterne alla base di conoscenza già costruita; Prolog non può eseguire correlazioni od assumere conclusioni circa cose che non gli vengono esplicitamente riferite.

Formuliamo un quesito

Formulare un quesito significa accedere ad un goal; esso può essere, come abbiamo già detto altre volte, presente nel programma o inserito, interattivamente, dall'esterno. Per ora ci limiteremo a proporre quesiti semplici, rappresentati da un solo, univoco goal; si tratta della rappresentazione più grezza del classico «goal», ma per adesso è sufficiente così; impareremo, successivamente, ad espandere il range delle domande ed a rendere più efficiente la formulazione di un quesito.

Battiamo, pertanto, alla tastiera:

```
Goal:mangiare(carlo,cioccolata)
```

otterremo come risposta:

```
True
Goal:
```

vale a dire che il programma assicura che, in base alla sua base di conoscenza, Carlo mangia la cioccolata. Battiamo ancora, di seguito, una serie di domande, avendo risposte incontrovertibili:

```
Goal:mangiare(nora,cioccolata).
False
Goal:bere(olga,birra).
True
Goal:essere_dannosa(palestra).
False.
```

Si notino le risposte fornite, ed una conclusione che salga immediatamente agli occhi: la risposta True indica, sempre, la verità di un «goal», di un'affermazione; la risposta False, al contrario, non sempre afferma la falsità di un'affermazione; False vuol dire, per la precisione: «Non certo», intendendo ambedue le alternative di «Non vero» e «Non so». In altre implementazioni del Prolog la seconda affermazione può, invece, essere differenziata con la risposta:

```
no data
```

opzione questa, non disponibile in Turbo Prolog.

Una visione più generale del modello di costruzione di una base di dati

Una situazione come quella descritta in precedenza, per quanto piuttosto chiara e lineare, è pur sempre una situazione almeno abnorme. In altri termini, il programma, così come costruito, serve a ben poco, visto che, così come è organizzato, non ci è dato di sapere se il sistema ha o non informazioni circa i gelati al limone od una persona chiamata Cristina o Michele. In altri termini occorre avere a disposizione una strada più semplice per inserire o ricavare informazioni dal sistema senza che per questo l'utilizzatore debba conoscere a priori l'estensione, le caratteristiche ed il contenuto del database.

Ancora, potremmo avere bisogno di sapere quante e quali persone contenute nel database possiedono certe caratteristiche. Ad esempio, potremmo dover sapere chi è che ama andare in palestra. Dovremmo eseguire una serie di tentativi del tipo:

```
Goal:andare(italo,palestra).
True
Goal:andare(olga,palestra).
False
Goal:andare(carlo,palestra).
False
Goal:andare(luana,palestra).
True
```

e così via fino a raggiungere tutte le possibilità, ammesso di non dimenticarne qualcuna. A questo punto il solito avvocato del diavolo direbbe: «Perché tante complicazioni? Non è più semplice leggersi il listato?». Avrebbe ragione, se non esistesse un metodo ben più elastico ed efficiente per gestire le informazioni.

Il metodo di gestire, da parte del Prolog, dinamicamente le informazioni è legato all'uso delle variabili. Sotto questo nome vanno una serie di entità di respiro ben più ampio delle omonime costruzioni del Basic o del «C»; come al solito, però, partiamo dal più generale; vedremo poi le successive particolarità d'uso.

Nella maggior parte dei linguaggi il concetto di variabile è ben definito, anche se, per chi si avvicina per la prima volta all'informatica, si tratta di un concetto un tantino strano. Il principio della variabile è quello che considera la stessa come qualcosa da riempire, da «inizializzare»; in altre parole, una variabile è come un foglio di carta od una casella in cui può essere inserito un valore. Quanto in questa casella inseriamo qualcosa, usiamo dire che la variabile è «istanziata» (in Turbo Prolog si usa dire che la variabile è «vincolata»; il termine opposto e «libera», o «inistanziata»). Nella maggior parte dei linguaggi di calcolo, il programmatore può fornire direttamente alla variabile valori (la «inizializza»). Le modalità di assegnazione sono diverse: può accadere che l'inizializzazione avvenga attraverso un valore compreso nel programma stesso; altre volte è lo stesso programma che esegue il calcolo e provvede all'assegnamento. Ancora il programma può provvedere ad indicare alla variabile dove cercare il valore, come ad esempio nei casi di input da file di dati; ancora, infine, l'assegnazione può essere eseguita in maniera interattiva, da tastiera.

Questo procedimento di assegnazione di variabile, così (ci venga perdonato il gioco di parole) vario, in Turbo Prolog è molto più limitato; l'assegnamento del valore viene, nella più parte dei casi, eseguita dal programma stesso. In ogni caso "l'instanziazione" (che schifezza di termine!) è sempre un'operazione correlata ad una attività del calcolatore, al contrario di quanto avviene in altri linguaggi altrettanto interattivi.

Definizione delle variabili

Le regole per la definizione (vale a dire l'assegnazione del nome) di una variabile sono piuttosto semplici e seguono, in ogni caso, molte delle regole comuni ad altri linguaggi. Ogni nome è valido, purché iniziante con una lettera maiuscola: in molti dialetti (ma non in Turbo) è ammesso, come primo carattere, il segno di sottolineatura (underscore [_]), ma si tratta di una complicazione che, come vedremo tra poco, è opportuno non considerare. La lunghezza non ha grande importanza. Ecco il motivo per cui avevamo così accuratamente evitato l'uso delle maiuscole. Ogni qual volta Prolog incontra qualcosa iniziante con una lettera maiuscola, la interpreta e la mette da parte come variabile da riempire, assegnare, successivamente, quando il programma sarà in esecuzione.

Vi sembrerà sorprendente, ma abbiamo istantaneamente risolto il problema della ricerca precedente («A chi piace andare in palestra?»); basterà scrivere

```
Goal:andare(Chi,palestra).
```

(ricordando che è possibile, in ogni caso, inserire già direttamente nel programma il Goal, invece di attendere che il programma stesso ce lo chieda), ed avremo come risposta:

```
Chi = italo
Chi = luana
Chi = marcello
3 Solution
Goal:
```

Che cosa è successo? Semplice! Prolog ha interpretato [Chi] come variabile ed ha tentato, in base alla sua base di dati, di assegnare i possibili valori (tutti, tre) alla variabile stessa. Allo stesso modo avremo:

```
Goal:bere(nora,Che_cosa).
```

la risposta sarà una sola vale a dire

```
Che_cosa=birra
1 Solution
Goal:
```

Una domanda, forse un po' sempliciotta, ma giusta: che succede quando, in un Goal, inseriamo non più una ma due variabili?: che risposta avremo a:

```
Goal:mangiare(Chi,Che_cosa).
```

Semplice: in base a quanto contenuto nel nostro piccolo database, Prolog effettuà una permutazione di quattro elementi su due posti; il risultato sarà

```
Chi = andrea, Che_cosa = cioccolata
Chi = andrea, Che_cosa = budino
Chi = andrea, Che_cosa = meringhe
Chi = biagio, Che_cosa = cioccolata
Chi = biagio, Che_cosa = budino
...
```

e così via. In effetti, in questo caso, l'uso di due variabili per i due argomenti del Goal non ha fatto altro che confondere il problema, visto che Prolog non ha fatto altro che assegnare tutti i valori possibili a [Chi] e [Che_cosa]. In generale la situazione si ripete sempre, e si rivela quasi sempre inutile trasformare in variabili tutti gli argomenti di un Goal; diviene invece più significativo, talvolta, introdurre, per certe applicazioni, N-1 variabili rispetto al numero N degli argomenti. Altra domanda sempliciotta: che cosa succede di fronte ad una richiesta del tipo:

```
Goal: mangiare (Nome, Nome).
```

vale a dire, utilizzando in ambedue i casi la stessa variabile? Semplice: una variabile non può contenere più di un valore alla volta. Nel momento in cui al primo [Nome] viene assegnato il valore, la risposta non può essere che [False] in quanto è come se chiedessimo a Prolog di indicarci chi è che mangia *se stesso* in base alle informazioni che contiene; non vi pare? Potremmo avere una risposta negativa solo se, per assurdo, avessimo posto la clausola:

```
mangiare(budino,budino).
```

cosa che porterebbe alla risposta

```
Nome = budino
1 Solution
```

Le variabili anonime

Esiste, nel nostro linguaggio, una variabile di tipo particolare, chiamata, appunto, variabile «anonima». Essa è rappresentata da un segno di sottolineatura, isolato [_]. Il nome le è derivato dal fatto che essa non è istanziata da alcun valore; così come è, sarebbe effettivamente priva di senso e valore pratico, se il suo uso non fosse insostituibile nell'uso delle strutture e delle liste.

Al contrario delle variabili definite per nome, le variabili anonime non mostrano il loro valore; infatti, non essendo mai istanziate, non hanno nulla da mostrare, visto, d'altro canto, che non possiedono alcun valore. Per quanto ci attiene finora, fin quando non arriveremo a parlare delle liste e delle strutture di dati, le variabili anonime sono praticamente esclusivamente usate nei luoghi dove è possibile andare incontro a variabili indesiderate o che potrebbero inquinare i dati di quelle necessarie al nostro studio, un esempio, anche se un po' forzato potrebbe essere derivato da quello precedente, in cui abbiamo così operato delle sostituzioni:

```
Goal:mangiare(_,Che_cosa).
```

(attenzione a non confondere l'underscore della variabile con quello compreso nel nome della seconda variabile). Il risultato è, ancora una volta, abbastanza semplice; sempre in base a quanto contenuto nel nostro piccolo database, Prolog risponde:

```
Che_cosa = cioccolata
Che_cosa = budino
Che_cosa = meringhe
...
```

e così via; i nomi delle persone non vengono mostrati, in quanto, al posto della variabile da istanziare, abbiamo inserito una variabile «dummy», fantoccio, tuttofare. È come se avessimo detto al Prolog: «Elencami tutte le cose che nel mio database, sono mangiate dalla gente».

Abbiamo così completato con le variabili; anche in questo caso Prolog si differenzia in maniera sostanziale dalle solite routine di definizione — inizializzazione — inserimento cui ci avevano abituato i più comuni linguaggi. Vedremo, la prossima volta, come Turbo Prolog maneggia le variabili nel suo interno. MC

AMIGA

AMIGA 500	**750.000**
MB 114 A (drive esterno per A 500)	**195.000**
AMIGA 2000 (1 drive)	**1.695.000**
AMIGA 2000 (2 drive)	**1.895.000**
MONITOR COMMODORE 1084	**495.000**
MONITOR PHILIPS 8802	**370.000**
MONITOR PHILIPS 8833	**470.000**

DIGITALIZZATORI: PRO SOUND DESIGNER **180.000**
FRAME SNAPPER **590.000**
(digitalizzazione video tempo reale)
HARDWARE E SOFTWARE: TUTTE LE NOVITÀ!

MB 286

L'AT SUPERLATIVO

MB 286/20 **2.490.000**
(512 Kb. 80286 10 MHZ o wait state, 2 seriali, 1 parallela, 1 disk drive 1.2 Mb, 1 hard disk 20 Mb, tastiera estesa)

MB 286/40 **2.990.000**
(come sopra ma con hard disk da 40 Mb/40ms)

MB 286/70 **3.490.000**
(come sopra ma con hard disk da 70 Mb/25ms)

COMBINAZIONI SCHEDA GRAFICA + MONITOR DISPONIBILI PER LA SERIE MB 286

HERCULES + MONITOR MONOCROM. ***390.000***
(14", schermo piatto, fosfori bianchi)

VGA + MONITOR/MONOCROMATICO **790.000**
(14", schermo piatto, fosfori bianchi)

VGA + MONITOR COLORI **1.490.000**
(14", schermo piatto, pitch 031)

ATARI

1040 STF	**850.000**
SM 124 (monitor monocromat.)	**230.000**
MEGAFILE 20 (hard disk 20 Mb)	**850.000**
PRO SOUND DESIGNER (digitalizz.)	**135.000**
MB 114 (disk drive 1 Mb)	**250.000**
MEGA ST 2/MEGA ST 4/LASER	**P.S.R.**

star

la tua stampante

LC 10 **490.000**
(144 CPS, NLQ, font residenti, paper park)

LC 10 C **590.000**
(come sopra ma a colori)

LASER PRINTER 8 **P.S.R.**
(Canon engine, 8 pag. min., 1 Mb RAM, 2 font)

NOVITÀ

LC 24/10 ***790.000***
(versione a 24 aghi della fantastica LC 10 - 160 CPS)

AMSTRAD

PC 1640 MD con HD 20Mb	**1.890.000**
PC 1640 ECD con HD 20 Mb	**2.490.000**
AMCARD 32 (hard disk card 32 Mb)	**890.000**
DRIVE 3.5" per PC 1640	**250.000**
PPC 512/PPC 640	**P.S.R.**

SOFT center *TUTTE LE NOVITÀ SOFTWARE PER TUTTI I COMPUTER*

TELEFAX

delle migliori marche **a partire da 1.695.000**

MEGABYTE

VENDITA ANCHE PER CORRISPONDENZA
PREZZI IVA ESCLUSA

UFFICI: VIA CASTELLO, 1 - 25015 DESENZANO del Garda (Bs) - Tel. 030/9144880
SHOWROOM: PIAZZA MALVEZZI 14 - 25015 DESENZANO del Garda (Bs)

MB