di Raffaello De Masi

Elementi di Prolog

nona parte

La costruzione di un programma

In questa puntata e nelle prossime vedremo, passo passo, come costruire un programma completo in Prolog. Inizieremo individuando che cosa la nostra base di conoscenza dovrà contenere, successivamente identificheremo una struttura, un sistema, per implementare efficientemente il tutto in un programma che tragga i migliori risultati dalle caratteristiche del linguaggio. Infine proveremo il programma per vedere come esso è capace di rispondere a domande. La conclusione sarà quella di vedere all'opera i tool necessari a costruire una base di dati capace di manipolare conoscenze, statiche o dinamiche, pronte a risolvere problemi reali. Ma questo rappresenterà solo un avvio; saremo poi capaci di spostarci verso operazioni più complesse con la tecnica acquisita, in modo da utilizzare (ed eventualmente creare) tool nuovi e più efficaci per rendere più efficiente, pratica e piacevole, la nostra tecnica di programmazione

Sembrerà strano quello che diremo, ma la cosa più complessa, e quella su cui cadono fior di programmatori abituati a pensare in altri linguaggi, è quella della organizzazione della base di conoscenza; sia ben inteso, niente impedisce di elencare così come, per esemplificare, abbiamo fatto finora, tutti i bit di conoscenza, uno appresso all'altro, e lasciare al programma il compito di dipanare matasse lunghe migliaia di righe; ma i motivi che sconsigliano tale pratica sono non pochi. Primo, esiste una dignità professionale, che impedisce di scrivere uno spaghetti-programma in Prolog, oggi che neppure in Basic o in Fortran avviene più; secondo, la risposta più ovvia è che se uno deve battere alla tastiera tale messe di dati impiegherebbe tanto di quel tempo che potrebbe proficuamente risolversi il proprio problema a mente, senza affannarsi dietro ad un programma da testare, spulciare e così via; terzo, una massa enorme di dati da inserire in un programma porta inevitabilmente all'errore, che può essere semplice (ortografia) ma anche subdolo (ve lo immaginate che cosa succederebbe in una base di conoscenza dove i predicati principali sono «porta» e «parte», quando la serie di record sono, ad esempio, un migliaio?). Bene, esistono alcune semplici tecniche per definire, nella maniera più pratica possibile, la base di dati ottimale per lo scopo che intendiamo perseguire. In accordo con le regole comportamentali proprie del Prolog, il sistema migliore per costruire un programma efficiente è quello di lavorare secondo schemi razionali, cosa facilitata, tra l'altro, dalle caratteristiche stesse del linguaggio.

Come in un ragionamento umano, nella redazione di un programma è fondamentale tener conto di tre direttive principali, che sono così riassumibili, in tre domande:

a) Quale è lo scopo del dominio di conoscenze che desidero organizzare?
b) Quali, tra i milioni possibili, sono i dettagli da conservare necessari per pervenire ad una risposta esauriente?
c) A quali specifiche richieste desideriamo che la base di conoscenza risponda per poter dare la soluzione che desideriamo?

Cercheremo di seguito di dare risposta a queste domande, non tanto in senso concettuale, visto che risposte in tale direzione se ne potrebbero dare a centinaia, ma in senso pratico; cercheremo cioè di evitare di caricare con orpelli e conoscenze inutili il nostro data base, in modo da evitare di portarsi a rimorchio spazzatura che all'atto pratico impiccia e rende più lento e faticoso il lavoro dell'elfo che corre nella nostra base di conoscenza.

Prendiamo ad esempio, come nostra prossima base di conoscenza, l'insieme degli scrittori del nostro secolo. Lo scopo è quello di organizzare una base di dati relativi alle caratteristiche biografiche e bibliografiche di tali autori. Ma così andiamo veramente male; di notizie di tal genere su tale argomento ce n'è da riempire dieci hard disk; e per battere tutte le notizie ci vorrebbero un paio d'anni. Una base di conoscenza così preparata sarebbe estremamente impratica, difficile da usare, lenta da consultare e, forse, inutile. Bene, dobbiamo per forza di cose limitare il nostro campo d'azione; è probabile che, alme-

Figura A - Costruzione di una base di conoscenza circa un argomento letterario.

Il Colombre è stato scritto da Dino Buzzati
Il Colombre ha 449 pagine
Il Colombre è un libro di racconti
Il Colombre prende il nome dal primo racconto della raccolta
Il Colombre è sttao pubblicato da Mondadori
Il Colombre è inserito nella collana "Scrittori italiani e stranieri"
Il protagonista del racconto "Il colombre" si chiama Stefano Roi
Il colombreé n mostro mitologico.
Il Colombre si compone di 51 racconti
Roberto Saggini è il protagonista del racconto "Cacciatori di vecchi"
Viale Regina Margherita, ne Il Colombre, è nominato tre volte.
La lunghezza media dei racconti, ne Il Colombre, è di quatttro pagine
Il Colombre è stato pubblicato nel 1966
Il Colombre è sttao ripubblicato sei volte
Ne Il Colombre esiste un'ultima parte, di sapore autobiografico
Ne Il Colombre vengono nominate 15 volte delle date.
Il Colombre è un libro senza presentazione
Il Colombre è, in tutte le pubblicazioni, redatto in sedicesimo
Il Colombre è scritto in caratteri Times, 10 punti tipografici
I titoli dei racconti ne Il Colombre sono scritti in copro 12
.....
(e così via)

a)	iniziare il programma con la sezione dei [Predicati]; definire i predicati di cui il sistema ha bisognoper conservare le informazioni inserite e per restituire i dati di cui si fa richiesta.
b)	elencare i predicati e gli argomenti presenti nella sezione [predicati], e determinare se siano stringhe, numeri, ecc. definire questi nella sezione [Domini] del programma
c)	inserire direttamente la base di conoscenza nella sezione [Clausole]
d)	concludere, ove necessario, con l'introduzione dei [Goal]

Figura B - Struttura organizzativa di un programma in Prolog.

no per la prima volta, avremo bisogno solo dei dati coinvolgenti un paio di autori, poniamo Umberto Eco e Dino Buzzati. La nostra base di conoscenza sarebbe, adesso, ragionevolmente manipolabile e, soprattutto, pratica (visto che, alla successiva occorrenza, niente impedirebbe di ampliarla con un nuovo autore). In altri termini, abbiamo organizzato la base di dati introducendo una limitata quantità di informazioni circa un ridotto numero di autori.

Ma anche così, sebbene limitate, le nostre basi di conoscenza, se volessimo introdurre tutti i dati relativi a Eco e Buzzati, sarebbero immense e, comunque, trabocchevoli di dati inutili, almeno per i probabili scopi per cui ci siamo messi alla tastiera. È quindi più opportuno delimitare certi campi specifici, in modo da rendere, da una parte più precisamente limitato il contenuto del DB, dall'altro ottenere scopi, per questo stesso motivo, più efficienti e finalizzati. Continuare in questo modo sarebbe come ritrovarsi con le mani piene di cose inutili. Potrebbe, ad esempio, essere necessario lavorare solo su uno o due opere di questi due autori. Questo ci consentirebbe di limitare il numero dei fatti da conservare nel sistema. Bene; costruiremo in questo caso, una base di conoscenza per un libro di questi autori, in modo da costruire un esempio di «massimo delle conoscenze circa un solo argomento». Un esempio di inizio di costruzione di data base come quello esemplificato è presente in figura A. In questa chi scrive non ha certo voluto dare prova di sue grandi conoscenze letterarie, anche perché l'ha fatto con il libro in mano; ha voluto solo dimostrare la molteplicità dei tipi di informazione inseribili, e come sia possibile, attraverso Prolog, conservare informazioni tanto diverse, con tecniche molto simili e facili da usare.

Tanto per intenderci, vogliamo ora, per gli scopi che ci siamo proposti, costruire una base di dati relativa alle opere di questi due scrittori; titoli e tipo di opera (novella, racconto, libro, commedia, saggio, ecc.), anno di pubblicazione, editore, autore, edizioni e loro tipo, ecc.

A questo punto dobbiamo compendiare le due opposte esigenze, presentate precedentemente. Occorrerà integrare ambedue le richieste per rendere più articolato il blocco di ricerca delle conoscenze. Come fare?

La cosa più semplice sarebbe quella di costruire una serie di istruzioni cui affidare la soluzione delle esigenze dell'utente della base di dati. Metodo semplice e facile da implementare, ma non pratico; è più opportuno, invece, costruire alcune regole [«rule»], che se ben redatte, semplificheranno in maniera notevole tutta l'architettura del programma.

Continuando con l'esempio appena descritto avremo il problema di decidere come e che cosa chiedere alla base di conoscenza. Tanto per usare una base comune e limitare l'esempio ad un blocco ben noto di dati, stabiliremo di organizzare una base di dati, relativa ad Eco e Buzzati, di dodici opere (sei per l'uno, sei per l'altro), con informazioni relative alla data di pubblicazione, autore, tipo ed editore delle dodici opere; questo condurrà ad inserire 48 informazioni [«fatti»] diversi, e due regole; roba di battitura, alla tastiera, in qualche minuto. Bene, individuata la struttura del programma, il disegno di base, iniziamo a costruire la base di dati; il procedimento che seguiremo, sebbene non rappresenti il massimo dello stile di redazione di un programma, è abbastanza efficiente e rappresenta il miglior compromesso tra facilità di redazione e velocità dei risultati. Fin tanto che non saremo abbastanza pratici da lasciare la strada maestra del Prolog, per cercare scorciatoie più efficienti (ma faticose), lo schema di redazione dei programmi che adotteremo da questo momento in poi può essere senza problemi usato per la redazione di qualunque programma.

La regola d'oro, anzi la serie di regole per la redazione di un buon programma in Prolog, sono descritte in figura B. Si tratta di uno schema semplice, facile da ricordare, seque1nziale, e che fornisce sempre buoni risultati, almeno fino a quando, come dicevamo precedentemente, non si sarà abbastanza scaltriti nell'uso dei più avanzati tool del linguaggio. Un altro approccio al problema è anche rappresentato dalla creazione di predicati, inizialmente, poi dall'immediato inserimento della base di conoscenza, e, infine, della definizione delle regole-istruzioni. Ambedue gli approcci sono egualmente efficienti, ma quello schematizzato in figura è, probabilmente, più logico e consequenziale, almeno secondo la nostra mente di umani.

Completiamo, ancora una volta, il discorso a metà, per proseguirlo la prossima volta con le regole di descrizione delle parti del programma; inizieremo con i predicati, per poi passare ai domini, ai fatti, alle regole e così via. MC

LE PERIFERICHE

DFI SCANNER L. 450.000
400 dpi - ora disponibile software OCR

STAMPANTI

Panasonic tutti i modelli	telefonare

MONITOR

monocromatico dual flat screen	L. 220.000
monocromatico VGA	L. 285.000
monocromatico multisync	L. 450.000
colori Philips 8802 (Amiga/ST)	L. 340.000
colori Philips 8833 (CGA)	L. 450.000
colori Philips 9043 (EGA)	L. 510.000
colori multisync CTX (nuovo)	L. 850.000
Mitsubishi Diamond Scan 1481	telefonare
schermi antiriflesso	da L. 18.500

SUPPORTI DI MEMORIZZAZIONE

chip RAM	telefonare
dischi 3,5" Precision	L. 2.000
dischi 5,25" Precision	L. 900
dischi 5,25" Precision HD	L. 2.300
drive 5,25" 1.2 Mb	L. 175.000
drive 3,5" 720 Kb	L. 180.000
drive 3,5" 1.44 Mb	L. 220.000
hard disk Seagate 20 Mb	L. 380.000
hard disk Seagate 32 Mb	L. 550.000
hard disk Seagate 40 Mb	L. 680.000
hardcard 20 Mb Tandon	L. 590.000
hard disk 100 Mb 18 ms con controller ESDI	
Adaptec interleave 1:1	L. 1.520.000

ADD-ON

coprocessore Intel 8087-5	L. 210.000
coprocessore Intel 80287-8	L. 480.000
coprocessore Intel 80287-10	L. 550.000
FAX Murata manuale italiano	L. 1.400.000
modem Smartlink esterno	da L. 195.000
modem Smartlink interno	da L. 180.000
mouse Z-nix 250 dpi	L. 85.000
tastiera 102 tasti Cherry	L. 110.000
tavoletta grafica Genius 12"	L. 750.000

SCHEDE

scheda copy card 4.5	L. 150.000
scheda eprom burner 4 pos.	L. 240.000
schede espansione memoria	telefonare
schede multifunzione XT/AT	telefonare
scheda Super EGA 640x480	L. 290.000
scheda Super EGA 1024x480	L. 330.000
scheda VGA 800x600	L. 450.000

130 tipi diversi di schede, accessori & add-on disponibili: richiedere il catalogo o telefonare!

I PERSONAL

tutti i tipi di cabinet:
desktop standard
desktop baby
desktop minibaby
trasportabili LCD
tower drive vert.
tower drive oriz.
minitower

PC XT 8088-10 desktop
512 Kb RAM espandibili 1 Mb
drive 360 Kb + hard disk 20 Mb
Hercules - tastiera 102 tasti
monitor 14" dual flat screen
Lire 1.600.000

PC AT 80286-12 desktop
512 Kb RAM espandibili 4 Mb
drive 1.2 Mb + hard disk 20 Mb
Hercules - tastiera 102 tasti
monitor 14" dual flat screen
Lire 2.100.000

PC 80386-20 tower (foto)
1 Mb RAM espandibile 8/16 Mb
drive 1.2 Mb + hard disk 32 Mb
Hercules - tastiera 102 tasti
monitor 14" dual flat screen
Lire 4.700.000

MODELLI BASE
assembliamo configurazioni su richiesta

LE NOVITÀ
FANTASOFT

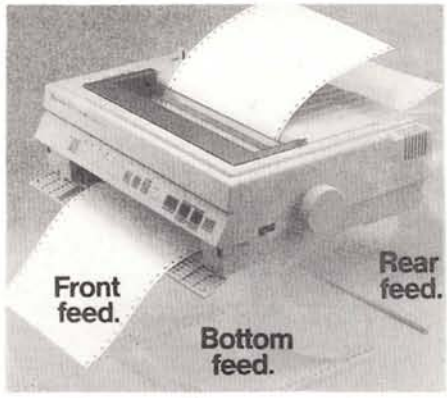
MOTHERBOARD 80286-12 MHz
finalmente disponibili le nuove piastre madri AT NEAT con:
- memory interleaved
- shadow ROM
- EMS 4.0 in hardware
- espandibilita' 4 Mbyte RAM

Lire 495.000
motherboard SUNTAC EMS 4.0 Lire 450.000

CHIP RAM 1 MBIT-100
piena disponibilita'

SCHEDA ESPANSIONE 2 MB
memoria EMS 4.0 2048 Kb
memoria estesa 2048 Kb
memoria DOS (512->640 Kb)
2 Mb 100 ns installati
prezzo incredibile! Lire 990.000

PANASONIC 1124
200 cps - 24 aghi telefonare



FANTASOFT
COMPUTER HOUSE

Via O.Targioni Tozzetti 7/b - 57126 LIVORNO
TEL: 0586/805.200 - FAX: 0586/803.094

PREZZI IVA E TRASPORTO ESCLUSI - RICHIEDETE CATALOGO - SCONTI A RIVENDITORI