di Raffaello De Masi

Elementi di Prolog

Le liste

«Ecco la grandine», disse Don Abbondio, «pochi chicchi ma buoni», narra il Manzoni, quando fa incontrare il buon pretucolo col cardinale Borromeo. La stessa cosa avranno pensato i lettori leggendo questo capoverso e pensando alle terribili liste del Lisp, con tutte le loro parentesi aperte o chiuse

In effetti una lista è solamente un tipo di struttura del Prolog (si vedano in proposito le considerazioni della volta scorsa), che permette manipolazioni facili, accurate e rapide di ampie messi di dati. Struttura comune al Logo e al Lisp (non per niente Lisp significa ListProcessing), possono essere intese, a tutti gli effetti, come uno strumento più generale e flessibile delle altre strutture.

La cosa più importante da tener da conto è che le liste hanno la loro importanza e funzionalità anche per il loro valore e la posizione di ogni elemento nella lista stessa. Altri linguaggi non specifici hanno fatto di quando in quando uso di liste, come ad esempio lo ZBasic, che possiede un array (Index$) dalle funzionalità molto prossime alle liste del Prolog e del Lisp; ma si tratta di episodi rari.

Un esempio semplice di lista è rappresentato già dalla struttura utilizzata le volte scorse; nella parola (lista) sono sottintesi già gli elementi che rappresetano una lista; così la sequenza (buzzati, feltrinelli, 1984, i girasoli) è tutta configurabile come una lista afferente al racconto «Il crollo della Baliverna». E un uso della lista l'abbiamo già visto quando abbiamo affermato che senza la corrispondente esatta sequenza il goal non poteva essere soddisfatto.

All'atto pratico, in una struttura che intende mettere insieme fatti collegabili univocamente a uno specifico elemento, l'approccio alla lista diviene efficace e efficiente. Ma cosa succede quando questa chiarezza di univocità non c'è più? Le liste non sono più utilizzabili, o magari esiste un tool, una tecnica più raffinata e precisa per manipolare liste anche in questa occasione? Non solo, ma cosa accade se la lista abbisogna di aggiunte e cancellazioni? La funzionalità di una lista tradizionale sarebbe in questo caso limitata. A questo difetto le liste del Prolog e del Lisp fanno adeguatamente fronte, soprattutto perché le liste non sono limitate in qualsiasi senso e non ammettono limiti, in ogni caso, che non siano logici.

Cosa è, in effetti una lista: la risposta potrebbe essere la seguente:

«Una lista è una sequenza ordinata, di lunghezza illimitata, di zero o più oggetti».

Due particolari importanti emergono da questa definizione; una lista è *ordinata*, e a questo ordine fa riferimento il sistema stesso. Tanto per capirci, due liste organizzate con gli stessi elementi, ma ordinate in maniera differente, sono considerate diverse a tutti gli effetti. Inoltre, una lista proprio perché di lunghezza infinita, comporta il fatto che può contenere da zero ad un numero infinito di elementi (si fa per dire, ovviamente, dato che il numero degli elementi è diretta funzione della capacità di memoria del computer stesso).

Esemplifichiamo il nostro dire; innanzi tutto premettiamo che una lista, in Turbo Prolog, è indicata da parentesi quadre:

```
testo([a,b,c,d])
animali([uccello,anfibio,ippopotamo])
uccello([rondine,struzzo,gallina,pettirosso,
passero,falco,corvo,beccaccia,cinciallegra]).
```

Notiamo subito che queste liste sono tutte formate da fatti, entità vere che restano sempre tali, visto che non contengono alcuna istruzione di [if]. La prima riga, tradotta in italiano significa, tal quale:

Esiste una lista, dal nome testo, che ha i suoi elementi rappresentati dalle lettere [(a,b,c,d)].

Possiamo anche usare le liste direttamente nelle regole e nei goal. La cosa è più semplice e intuitiva di quanto si immagini e ne vedremo tra poco l'applicazione.

Organizzazione di una lista

Per convenzione una lista (tranne una eccezione), consiste di una testa e di una coda. La testa è definita come il primo elemento della lista, la coda è rappresentata da tutti gli elementi seguenti. Una rappresentazione di quanto stiamo dicendo è nella figura a, dove è esemplificata la struttura della lista evidenziata nell'esempio espresso in precedenza. Prolog definisce una lista senza elementi una lista vuota; essa non ha né testa né coda.

Occorre ancora fare una precisazione; la coda di una lista è essa stessa una lista; questo è vero anche quando la lista con cui si sta lavorando ha solo uno o due elementi. Ciò porta a delle particolarità; la coda di una lista con due elementi è una lista di un singolo elemento, e la coda di una a singolo elemento è una lista vuota. La figura b mostra quanto abbiamo detto in maniera più chiara, evidenziando le diverse possibilità di «consistenza» di una coda.

Definizione di una lista

In Turbo Prolog, le liste devono essere definite come accade nei domini, prima di essere usate o verificate. Le liste devono essere pertanto descritte, come [string], [integer], [real], [char] e [symbol].

testa	coda
a	b,c,d
uccello	anfibio,ippopotamo
rondine	rondine, struzzo, gallina, pettirosso, passero,falco,corvo,cinciallegra

Figura a
Le parti di una lista.

Vediamo subito come vanno definite alcune strutture.

```
domains
testo=symbol*
animali=string*
uccello=string*
valore=integer*
conto=real*
```

Come si noterà, una lista è definita come un dominio aggiungendo un asterisco alla fine della specificazione di tipo. L'asterisco significa, in altri termini, «lista di...», e nel caso in esame,

```
testo=lista di symbol
animali=lista di string
uccello=lista di string
valore=lista di integer
conto=lista di real
```

Ovviamente è sempre possibile dichiarare una lista come un tipo di dominio, ma questo era implicito nella dichiarazione stessa. La vera grande limitazione, implicita nel Turbo Prolog è che una lista può possedere uno e un solo tipo di dato. È possibile, con un artificio, superare questa limitazione, ma anche così la dichiarazione di una lista costituita da diversi oggetti è pur sempre un affare serio e tedioso. D'altro canto c'è da dire, a conforto di questa situazione, che ben poche volte si presenta l'occasione di dover ricorrere a questi trucchi.

Una volta che i domini sono poi stati ben definiti In Turbo Prolog, le liste possono essere utilizzate nella maniera più elastica ed efficiente possibile.

Ancora c'è da dire una cosa, sulla dichiarazione delle liste; poiché, per definizione, in Prolog una lista può essere rappresentata da argomenti di qualsivoglia genere, è logico ammettere che uno o più elementi di una lista possano essere essi stessi delle liste. L'affermazione è più chiara se guardiamo gli esempi seguenti:

```
lista([a,[b,c],d,e,[f,g,h,[i,l,m]]])
animali([cane,[mosca,zanzara,farfalla],[leone,
tigre,lince]])
```

rappresentati graficamente in figura c. La lista principale è rappresentata, nel primo caso, dagli elementi [a], [d], [e] cui segue la disposizione delle secondarie, correlate ([b,c]) ad [a], ([f,g,h,i,l) ad [e]. Nel secondo caso le cose sono più intuitive, con due subliste ben individuate, collegate alla lista principale [cane].

La manipolazione delle liste è uno degli strumenti più potenti ed efficienti in Turbo Prolog. Sulle liste è possibile

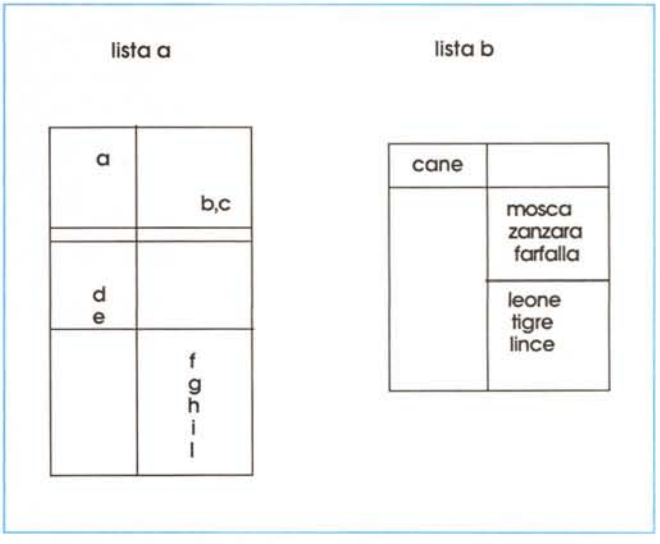
lista	coda
(a,b,c,d)	(b,c,d)
(a,b)	(b)
(a,b,(c,d))	(b(c,d))
( )	*nessuna*
(n)	( )

Figura b - Individuazione delle code.

costruire predicati, sviluppare goal anche complessi, specie con gli operatori numerici visti in precedenza, e ancora manipolare basi di dati ben estese e strutturate. Purtroppo, per ragioni non ben chiare, molte versioni di Prolog mancano di operatori specifici sulle liste, come quelli, tanto per intenderci, presenti in Lisp. Si tratta di una mancanza talora pesante, che obbliga il programmatore a veri e propri funambolismi per giungere a risultati concreti e efficaci. Fortunatamente, tutti i libri, dedicati a Prolog, evidenziando la mancanza e la necessità di costruire tali strumenti, forniscono utility già pronte per bypassare il problema e risolvere con soddisfazione la bisogna.

Ma prima di avventurarci in queste utility, che cercheremo, nel nostro piccolo, di evidenziare in maniera il più chiara possibile (cosa che faremo la prossima volta), occorre impadronirsi di una tecnica, la ricursione (o ricorsività) senza la quale tutti i nostri bei discorsi vanno a buone donne.

Figura c
La correlazione tra liste e subliste.



Ci riserveremo la prossima volta di ampliare il discorso con esempi più chiari.

Oggi cercheremo di dare solo una idea di base del processo.

Premesso che il concetto di ricursione, per uno di quegli strani giochi della vita che ci capita di affrontare ogni giorno, può risultare addirittura ostico a certe persone, credo che la migliore definizione di questo termine l'abbia data D. Shafer, nel suo eccellente volume sul Prolog: la ricursione può essere così definita:

ricursione, s.f. v. ricursione

A parte il paradosso della definizione stessa, occorre chiarire che con ricursione si intende (per chi già non lo sapesse), una struttura, uno statement, che per essere eseguito chiama se stesso, figuratamente è come dire che la pioggia è la cosa che cade quando cade la pioggia, o come tentare di sollevare l'ascensore tirando dalla cabina le corde dell'ascensore stesso. Concetto non facile da intendere e, sovente, ancor meno da utilizzare da parte di chi non ne ha afferrato l'intima essenza, la ricursione è uno degli strumenti più potenti e versatili della programmazione moderna.

In Prolog una struttura ricorsiva (ed ecco qui il motivo per cui ne parliamo adesso) esplora senza difficoltà le liste, molto più facilmente e rapidamente che con un programma costruito ad hoc. Ma è tempo di chiudere; continueremo il discorso la volta prossima; a risentirci!