L'Intelligenza Artificiale

di Raffaello De Masi

I linguaggi d'elezione dell'Intelligenza Artificiale: il LISP

Quinta parte

Ogni simbolo in Lisp può avere degli attributi associati con esso. Questi non sono altro che valori o rappresentazioni simboliche abbinate ai simboli stessi. In altre parole è possibile agganciare materialmente ad un simbolo una apposizione che lo qualifica. Tutto ciò viene eseguito con la funzione [putprop] che va usata nel modo:

(PUTPROP *simbolo valore attributo*)

L'informazione può essere attinta usando [get] nel modo:

(get *simbolo attributo*)

La figura A mostra come è possibile eseguire assegnazione di attributi e valori a simboli, e come questi possono essere ispezionati.

Una volta assegnato il valore, ad un attributo, questo resta finché non viene modificato (si veda ancora la figura A) o viene eseguita una rimozione dell'attributo nel modo

(remprop *simbolo attributo*)

Puntatori e locazioni di memoria

Consideriamo adesso l'esempio di figura B. In esso si esegue la già vista assegnazione dell'attributo colore al simbolo ferrari. Dopo di ciò, col [setquote] viene assegnato alle variabili «auto 1» e «auto 2» il medesimo valore «ferrari». In altri termini le due variabili puntano alla stessa struttura dati ed infatti, come si vede, l'operazione di get, applicata ad ambedue le variabili [auto 1] ed [auto 2] fornisce lo stesso risultato. O per dirla alla Lisp esiste un solo simbolo, [ferrari], e due variabili, [auto 1] ed [auto 2], ambedue puntanti allo stesso valore, con una rappresentazione visualizzabile come nella seconda parte della figura. Ogni simbolo, in Lisp, possiede una sua precisa locazione in memoria, e tutti i riferimenti ad esso non sono altro che puntatori alla stessa, unica, locazione. Questo vuol dire che, quando Lisp analizza un simbolo, si accerta se esso è già inizializzato (determinato, creato, o, comunque, in possesso di sue caratteristiche ben specifiche) e, nel caso ciò sia, non ne crea una nuova.

Quello che avviene per i simboli, gli atomi, non è invece sempre vero per le liste. Due liste possono essere identiche ma far riferimento od essere allocate in differenti aree di memoria. In altri termini, in presenza di una nuova lista, Lisp creerà immediatamente una nuova struttura del tutto indipendente, senza preoccuparsi se e quando esista o sia mai stata formata una lista con le stesse caratteristiche. Per intendere meglio il concetto si guardi la figura c; [auto A] ed [auto B] puntano a differenti liste della struttura.

Invece [auto C] punta alla stessa struttura di [auto A] poiché il valore di [auto C] viene in pratica settato a quello di [auto A] e questo rende il puntatore di [auto C] del tutto identico a quello del precedente.

A proposito di ciò Chamiak e Mac Dermott, autori di «Artificial Intelligence», il libro ispiratore di molte delle note presenti in questi articoli, e da cui ho tratto la gran parte degli esempi fin qui discussi (una esauriente bibliografia sarà proposta alla fine di questa serie di note) pongono un non banale quesito di filosofia di Lisp.

La domanda è: «[auto A] ed [auto B]» sono la stessa cosa?». La risposta è si o no; nel primo caso, in effetti, le due liste contengono elementi uguali, nel secondo caso, invece contengono gli stessi elementi o, il che è lo stesso, sono guidate dallo stesso puntatore.

A questo stato di cose corrispondono due funzioni [equal] (che d'altro canto abbiamo già visto) ed [eq.]. Ambedue si utilizzano allo stesso modo vale a dire:

(equal espressione1 - espressione2)

(eq espressione1 - espressione2)

ma mentre nel primo caso viene creata una nuova lista ed [espressione1] ed [espressione2] possiedono aree di memoria e di immagazzinamento dati distinte, ancorché conservati identici dati, nel secondo esse fanno riferimento ad un'unica area dati, guidate come sono dallo stesso puntatore.

Da quanto appena detto discendono diverse conseguenze tutte esemplificate nella figura D.

Gli stessi autori, appena accennati, evidenziano come la forma di figura C sia una versione molto semplificata dalla vera struttura interna del linguaggio. In pratica ed in maniera più accurata, anche se non certo precisa in assoluto, occorre affermare che, in Lisp, le liste sono conservate in forma concatenata tra di loro, vale a dire che ogni elemento di esse punta al successivo tranne l'ultimo che punta a [nil] (un end-of-file sui generis). In pratica è possibile considerare ogni atomo di una lista composto di due parti, la prima contenente il valore dell'atomo stesso, la seconda contenente il puntatore al successivo membro della lista. Nel caso di subliste, la prima parte non contiene dati, ma un ulteriore puntatore alla sublista stessa, e cosi via.

Stiamo avvicinandoci al termine della nostra pur superficiale e per forza di cose rapida trattazione del linguaggio Lisp. Dedicheremo la prossima puntata ad alcuni particolari aspetti del loop ed ad alcune funzioni di I/O. Poi passeremo ad un nuovo aspetto dell'IA che avrà bisogno di quanto abbiamo finora esposto per poter essere trattato.

```
> (putprop 'ferrari 'rossa 'colore)
□ (rossa)
> (putprop 'ferrari '(maranello modena) 'luogo)
□ (maranello modena)
(abbiamo assegnato due proprietà a l'ferrari) ; in un primo tempo (rossa) ,
come attributo (colore) , e successivamente la lista stessa ( maranello
modena) come attributo [luogo] .)
>(get 'ferrari 'colore)
□ rossa
>(get 'ferrari 'luogo)
□ maranello modena
>(putprop 'ferrari 'gialla 'colore)
□ gialla
(abbiamo assegnato all'attributo [colore] il valore [gialla])
>(get 'ferrari 'colore)
□ gialla
(ed il valore [gialla] prende il posto di [rossa] della precedente
assegnazione)
```

Figura A

```
>(putprop 'ferrari 'rossa 'colore)
□ rossa
>(setq auto1 'ferrari)
□ ferrari
>(setq auto2 'ferrari)
□ ferrari
>(get auto1 'colore)
□ rossa
>(get auto2 'colore)
rossa


---------


auto1 \
       \
        'ferrari 'rossa 'colore
       /
auto 2 /
```

Figura B

```
>(setq autoa '(ferrari maserati lamborghini))
□(ferrari maserati lamborghini)
>(setq autob '(ferrari maserati lamborghini))
□(ferrari maserati lamborghini)
>(setq autoc autoa)
□ferrari maserati lamborghini)
```

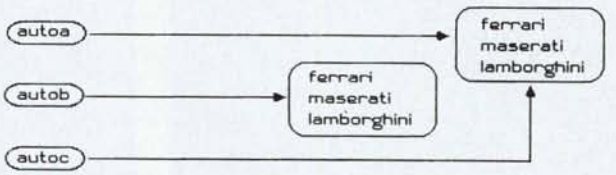

Figura C

```
>(eq autoa autob)
□ nil
>(equal autoa autob)
□t
(nel primo caso il risultato è falso in quanto i puntatori sono diversi ed
indirizzano a differenti locazioni di memoria ; nels econdo il confronto
immediato dei contenuti di due arre di memoria , anche diverse , porta a
risultato [t] , "vero" , in quanto i contenuti sono effettivamente eguali)

>(eq 'autoa 'autoc)
□t
------------
>(eq 'auto 'auto)
□t
(ogni atomo è [eq] , e conseguentemente [equal] a sé stesso)

>(eq 'autoy 'autoz)
□nil
>(setq auto1 (list 'ferrari))
□(ferrari)
>(setq auto2(list'ferrai))
□(ferrari)
>(eq auto1 auto2)
□nil
>(equal auto1 auto2)
□t
(il che dimostra che quanto appena detto per gli elementi è valido anche per
le liste)
```

Figura D

DISI

PRE
IL PC COMPATIB
A L. 9

DISITACO s.r.l.

DIREZIONE SERVIZI COMMERCIALI
Sede operativa: Via Arbia, 60
c.a.p. 00199 Roma Italia
Tel. 06/84.40.766 - 86.77.41

PUNTO VENDITA DISITACO S.R.L.
Via Massaciuccoli, 25/A
c.a.p. 00199 Roma Italia
Tel. 06/83.90.100

GRUPPO VENDITA DISITACO
vedi pagine gialle
voce *Personal computers*

- **PC TURBO 1024K**

  RAM 1024K - CLOCK 4.77/8 MHz - 8 SLOTS
  1 DISK DRIVER 360KB - ALIMENTATORE 150W
  SCHEDA GRAFICA - TASTIERA EVOLUTA

  L. 999.000

- **PC TURBO 1024K**

  RAM 1024K - CLOCK 4.77/8 MHz - 8 SLOTS
  2 DISK DRIVER 360KB ALIMENTATORE 150W
  SCHEDA GRAFICA - TASTIERA EVOLUTA
  MONITOR MONOCROMATICO

  L. 1.599.000

TACO

SENTA
LE TURBO DA 1024K

09.000

- **PC XT TURBO 1024K**

RAM 1024K - CLOCK 4.77/8 MHz - 8 SLOTS
1 D. DRIVER 360 KB - 1 HARD DISK 20MB
ALIMENTATORE 150 W - SCHEDA GRAFICA
TASTIERA EVOLUTA - *MONITOR B/N*

L. 2.599.000

- **PC XT TURBO 1024K**

RAM 1024K - CLOCK 4.77/8 MHz - 8 SLOTS
1 D. DRIVER 360 KB - 1 HARD DISK 20MB
ALIMENTATORE 150 W - SCHEDA GRAFICA
TASTIERA EVOLUTA - *MONITOR COLORI*

L. 2.899.000

- **PC AT TURBO 1024K**

RAM 1024K - CLOCK 6/8 MHz - 8 SLOTS
1 D. DRIVER 1200 KB - 1 HARD DISK 20MB
ALIMENTATORE 200 W - SCHEDA GRAFICA
TASTIERA EVOLUTA - *MONITOR B/N*

L. 3.699.000

- **PC AT TURBO 1024K**

RAM 1024K - CLOCK 6/8 MHz - 8 SLOTS
1 D. DRIVER 1200 KB - 1 HARD DISK 20MB
ALIMENTATORE 200 W - SCHEDA GRAFICA
TASTIERA EVOLUTA - *MONITOR COLORI*

L. 3.999.000

- **REGOLARE LICENZA**
MS DOS e GW BASIC
della *Microsoft Corporation*

- **GARANZIA COMPLETA**
di 1 anno e contratti di assistenza pluriennali curati da *D.C.S. ITALIA*