di Raffaello De Masi

I Sistemi Esperti

Rosie, è ancora oggi, a distanza di tanto tempo (in informatica e, ancora di più, nei sistemi esperti, si fa presto a invecchiare, anche in pochi mesi), un punto di riferimento per chi decide di costruire sia un linguaggio di I.A., sia per chi ha intenzione di produrre un Sistema Esperto. I meriti li abbiamo elencati già diverse volte, e la sua struttura di base è fin troppo nota presso la maggior parte degli utenti di queste discipline. Oggi continuiamo nell'illustrazione di questo ambiente, evidenziando una serie di caratteristiche dello shell stesso che consentono di far ritenere Rosie una palestra del tutto efficiente anche per chi all'Intelligenza Artificiale si avvicina nelle vesti di non specialista.

La tecnica fondamentale di rappresentazione di Rosie

Il principio fondamentale di utilizzo di Rosie è basato sulla rappresentazione di una base di conoscenza propria di un esperto; la tecnica di sviluppo è quella di combinare una struttura, ancorché semplice, con un'altra e così via per formare strutture sempre più complesse. Il tutto, come nel miglior costume dei linguaggi di intelligenza artificiale, racchiuso in uno shell e rappresentato da basi di dati e da blocchi di regole (si ricordi che per definizione i database sono blocchi di conoscenza descrittiva, e le regole rappresentano conoscenza prescrittiva o procedurale). Merito di Rosie è stato quello di rappresentare quest'ultimo tipo di conoscenza come un gruppo di proposizioni primitive raggruppate in un database. Le proposizioni di cui questa conoscenza è composta sono rappresentate dalla descrizione iniziale delle condizioni del problema, dalle procedure e dai fenomeni inferenziali sviluppati nel corso del disegno del sistema esperto, e dal blocco delle conclusioni finali. Per essere più precisi, Rosie rappresenta la conoscenza prescrittiva attraverso un blocco di regole, espandibili, controllate da un monitor. In definitiva ogni modello costruito con Rosie consiste di uno o più database e di uno o più blocchi di regole.

Nonostante questo esteso panorama in cui si estende, Rosie possiede alcuni limiti intrinseci, in certi casi pesanti; sebbene un programmatore possa, in questo ambiente, creare strutture sia descrittive che prescrittive, il programma in sé potrà solo creare database. L'applicazione finale costruita con Rosie non potrà mai modificare o generare nuove regole, né, ovviamente, blocchi di esse. Le strutture fondamentali su cui Rosie si basa sono otto: elementi di base, forme relazionali di base, metaelementi (comprendenti proposizioni, descrizioni intenzionali, azioni intenzionali), database, database condivisi, blocchi di regole, demoni e monitor.

Gli elementi di base

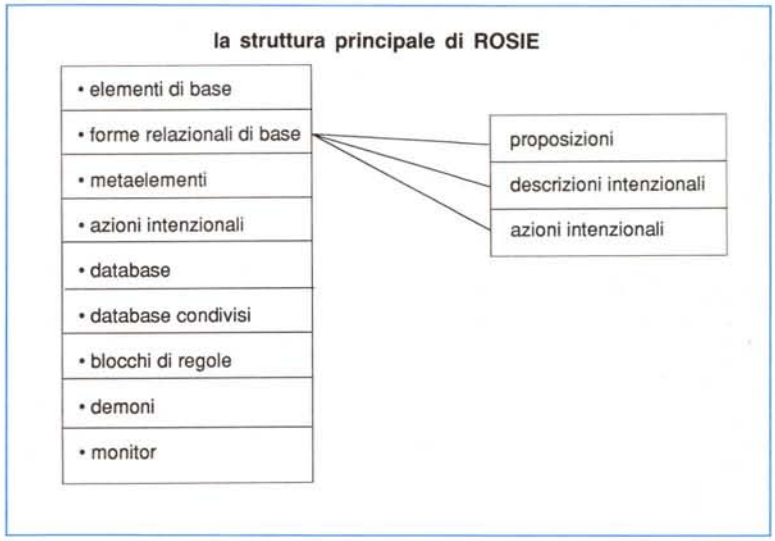
Rosie definisce i suoi elementi in termini di concetti. Questi elementi comprendono una serie di tipi di dati diversi: nomi, stringhe, numeri, combinazioni, classe di elementi, e pattern. L'elemento consente al programmatore di rappresentare nomi alfanumerici consistenti di una o più parole; ad esempio *Andrea, procedura n. 3*, o *rapido Napoli-Bologna*. Importante è la possibilità di utilizzare, come si vede, nomi composti da più parole per definire oggetti e concetti. La stringa è una sequenza di caratteri delimitata da simboli di riferimento (ad esempio le virgolette ["]). Si tratta di elementi piuttosto simili a quelli presenti in altri linguaggi, e, come in essi, esistono una serie di tool destinati alla concatenazione e alla ricerca di substringhe.

Rosie supporta tre tipi di elementi numerici: numeri semplici, costanti unitarie e costanti etichettate. Viene superata la sovente fastidiosa differenza tra numeri in virgola mobile e interi, e nei numeri semplici sono compresi numeri di tutti i tipi, interi e non, in qualsiasi precisione voluti. In questa categoria sono quindi inseribili allo stesso modo valori come [2], [3.1416], [126000456] o [6.2E23]. Numeri di questo tipo non hanno unità o etichette (nomi).

Le altre due classi sono uniche di Rosie e permettono di definire oggetti più complessi. Nel primo caso (costanti unitarie, ma sarebbe meglio dire costanti con unità) si tratta di valori numerici, seguiti dalla loro unità di misura; si tratterebbe di ben poca cosa, se non si tenesse conto del fatto che il linguaggio è tanto intelligente da saper manipolare anche le unità di misura. Ad esempio avremo che:

```
Display 34.5 KG/(25 CMx12 CM)
darà .115 KG/CM2
```

dove [Display] equivale al più comune Print di altri linguaggi (Rosie distingue, nella stampa, l'output su video da quello su altre periferiche). Infine le costanti con etichetta sono numeri preceduti da



una serie di caratteri, parole, token, insomma, che ne permettono l'individuazione. Un esempio è:

```
pigreco          3.14
avogadro         6.02E23
```

Unità ed etichette migliorano in maniera sostanziale la redazione e la leggibilità dei programmi; inutile insistere sulla possibilità di manipolare con operatori unità di misura e simboli.

Sotto il nome di combinazioni vanno liste di elementi organizzati in una singola struttura. Ogni membro di una combinazione può a sua volta essere rappresentato da un'altra combinazione. Un esempio è:

<matrice,<1,2,3>,<1,3,5>,<2,4,6>>

oppure

<padre, madre, figlio, figlia, nonno>

La classe di elementi è qualcosa di più; essa specifica un intero set di elementi, ed è rappresentato da una descrizione indicante quali elementi fanno parte di una classe. Ad esempio

tutti i numeri interi compresi tra 1 e 10
tutti gli alunni di nome Mario
tutti gli articoli giacenti in numero inferiore a 30 e che sono stati acquistati da più di due anni

Infine i pattern. Si tratta di un costrutto specifico di Rosie, e disponibile solo per operazioni di I/O o di confronto. Consiste di strutture estremamente specializzate, efficienti, che semplificano enormemente queste operazioni e, per estensione, quelle sulle stringhe. In questo caso la tecnica d'uso è piuttosto efficiente. Un pattern è rappresentato da una sequenza di subpattern racchiusi tra parentesi e separati da virgole. Ogni subpattern rappresenta a sua volta una restrizione del pattern da cui dipende, relativo a una successiva porzione della stringa di testo. Oltre queste strutture di base esistono i cosiddetti metaelementi (come proposizioni, descrizioni intenzionali, azioni intenzionali) che permettono di estendere i tipi di conoscenza che possono essere espressi, manipolati e rappresentati. Il principio che regola questi metaelemeti, che fanno comunque parte dell'area descrittiva, è basato sul fatto che essi manipolano e organizzano «intenzioni», invece di quantità fisse o variabili. È per questo motivo che occorre chiarire, almeno in larga massima, il significato di forma relazionale di base.

Cosa è una forma relazionale

Una forma relazionale è né più né meno una rappresentazione semplificata degli elementi di una analisi logica di una proposizione. Un costrutto, una regola, un elemento cognitivo sono divisibili in un massimo di 5 elementi costitutivi; elemento principale, predicati, complementi del predicato, verbi transitivi e verbi intransitivi. Un esempio di costrutto di tal tipo e degli elementi appena nominati è quello che segue:

```
elemento è un nome di oggetto
   il macintosh è un calcolatore
elemento è un aggettivo
   il macintosh è grigio
elemento è un complemento del predicato
   il macintosh è molto compatto
elemento è un verbo
   il macintosh pesa
elemento è un elemento del verbo
   il macintosh pesa poco
```

I costrutti non sono sempre semplici come quelli appena visti; ad esempio è possibile inserire negazioni anziché affermazioni («macintosh non è un cane») o graduare le affermazioni stesse («macintosh costa più di apple!»). È possibile anche estendere il numero degli elementi in una relazione aggiungendo valori accessori all'affermazione principale, come in

macintosh viene usato per il word processing
macintosh usa un monitor ad alta definizione
macintosh non piace a corrado

Le proposizioni inquadrano una relazione di base nella forma di un elemento. Formalmente Rosie individua una proposizione includendola tra due segni di virgoletta (singola ['']). Ad esempio:

'Marco è il direttore'
'Corrado scrive di giochi intelligenti'
'Raffaello è un pilota mancato'

sono tutte proposizioni. Un esempio d'uso di questo tipo di costrutto appare chiaro se si considera che nelle basi di conoscenza esistono dei costrutti-valori che non possono cambiare, o, per meglio dire, possono assumere solo il valore di sì o no; in questo caso è inutile affrontare il lavoro di costruzione di un motore inferenziale destinato a dare una risposta articolata, laddove una già precostruita darebbe lo stesso risulato. La possibilità di usare le proposizioni estende in maniera impensabile la potenza e la capacità delle prestazioni di Rosie. Questo soprattutto se abbinato al successivo tool, che di seguito mostreremo.

Descrizioni intenzionali

Una descrizione intenzionale è, tecnicamente, un riferimento implicito a una classe di elementi. Le descrizioni sono utilizzate dallo shell in associazione con un determinatore o un quantificatore per manipolare uno, alcuni, o tutti i membri

di una classe di elementi. Le descrizioni intenzionali, invece, forniscono un meccanismo attraverso il quale questa tecnica di accesso può essere sospesa o regolata a piacere o secondo il desiderio dell'utente. In un certo senso le descrizioni intenzionali funzionano come puntatori ai gruppi di elementi, un poco come fa la funzione «call-by-name» del linguaggio Algol.

Le descrizioni intenzionali sono delimitate all'inizio e alla fine, da singole virgolette come negli esempi:

'la lista degli attrezzi'
'il comando di stampa'

L'uso dei descrittori intenzionali permette agli ingegneri della conoscenza di rappresentare elementi indefiniti e concetti generici (ad esempio, il colore viola, la lista degli invitati, ecc.) e, per giunta, l'elemento esplicito riferito dalla stessa descrizione intenzionale può anche non esistere (come nel caso di elemento nullo). Con una tecnica molto simile a quella di altri linguaggi di I.A. una descrizione intenzionale avviene attraverso l'uso del comando [instance of]. Ad esempio [instance of attrezzi] darà come risultato la serie <chiave dinamometrica, chiavi a bocca, chiave a rullino, cacciavite a lama> e così via, a seconda del materiale presente nella specifica lista.

La tecnica del call-by-name presente in Rosie permette di eseguire anche chiamate intenzionali regolate da condizioni. Ad esempio è possibile aggiungere un elemento a una lista generica col sistema:

se l'auto da riparare è a motore wankel
aggiungere alla dotazione degli attrezzi un estrattore dei segmenti di tenuta

ovviamente dopo aver definito la struttura «dotazione degli attrezzi» nel modo opportuno.

Prima di concludere ci resta da parlare della azione intenzionale. Quest'ultimo metaelemento permette a un ingegnere della conoscenza di rappresentare una azione non ancora eseguita, e destinata ad essere sviluppata in un secondo tempo. Si tratta di una componente del linguaggio di uso molto esteso se si considera che, con essa, è possibile costruire sistemi che eseguono pianificazioni e le sperimentano; si tratta, in altri termini, di un mezzo ideale per eseguire simulazioni. Un'azione intenzionale determina l'uso di un imperativo; un esempio potrebbe essere:

Affondate la Bismarck
Creare una regola per il calcolo dell'area del rettangolo

Anche qui si può fare un uso abbastanza esteso e articolato dei complementi di condizione; così la regola precedente può essere modificata nel seguente modo:

Affondate la Bismarck se mostra intenzioni ostili e sventola la bandiera di guerra

I metaelementi appena descritti (proposizioni, descrizioni intenzionali e azioni intenzionali) rappresentano tre delle strutture più importanti dello shell Rosie. Costruita una proposizione, un programma scritto con Rosie può inserirla in un database, estrarla da esso o solo verificarne la sua presenza. La descrizione intenzionale conserva una rappresentazione, non ancora valutata, descrizione che può rappresentare un concetto indefinito o una particolare relazione tra gli elementi del database. Infine una azione intenzionale esegue un imperativo, valutato o non; si tratta di una tecnica molto raffinata che permette di pianificare, analizzare e dirigere azioni, anche in funzione di previsioni future sulle conseguenze delle stesse.

MC

Automobili e Sistemi Esperti

Continuiamo con la nostra aneddotistica relativa alle luci e ombre dell'uso della I.A. e dei S.E., uso che, è opportuno dirlo, diviene ogni giorno più diffuso e sviluppato. La volta scorsa abbiamo illustrato l'esempio di Thompson e l'utilizzazione della supertabella per il gioco degli scacchi per evidenziare come certe tecniche di semplificazione di problemi anche abbastanza complessi possano portare a una gestione poco «intelligente» del problema, con tutte le complicazioni connesse. Un esempio analogo può essere quello della adozione di un Sistema esperto piuttosto complesso che fu installato a Los Angeles, in California, finalizzato al riconoscimento e alla diagnosi di guasti in una grossa officina di una concessionaria della G.M. Dopo un uso piuttosto esteso si notò come il sistema tendesse ad attribuire a mancante o a errata lubrificazione certi guasti connessi a surriscaldamento dei motori, cosa che invece era solo dovuta a difettoso funzionamento di certa componentistica compresa in un dato lotto da parte di un fornitore esterno. I costruttori del S.E. avevano implementato nel loro sistema la possibilità di diagnosi di cattivo funzionamento di strumentazioni di controllo, ma mancava alla macchina qualunque sensibilità relativa a considerazioni del tipo «Se in questo periodo si sono intensificati i guasti relativi all'impianto di lubrificazione, non potrebbe essere per qualche guasto a monte della lubrificazione stessa?». Il sistema, invece, verificava le cause dei possibili guasti (pompa difettosa, filtro otturato) senza comunque tener conto di una possibilità, forse meno plausibile, ma comunque possibile, che poteva essere rappresentata da un difetto di fornitura. Si potrà obiettare che una successiva modifica del sistema potrebbe comprendere le modifiche apportate da quella esperienza. Ed è proprio qui il problema; il sistema non è provvisto di capacità di imparare, da una parte, né (come abbiamo fatto notare diverse volte negli articoli precedenti) ha possibilità di analizzare problematiche in maniera diversa da quanto gli hanno insegnato attraverso le famigerate «tabelle» di cui già dicemmo la volta scorsa.

Esistono già oggi le possibilità di disporre di memorie di migliaia di gigabyte; la tecnica più bruta di soluzione dei problemi porta alla inevitabile tentazione di creare una serie interminabile di domande con una ancora più inteminabile serie di risposte. Tutto può essere riassunto nella creazione di grosse banche dati di domande-risposte relative a una vasta gamma di campi di applicazione, ovunque esistano problemi che abbisognano di soluzione. Nonostante tutto ciò possa sembrare un vantaggio, ciò può risolversi in un grave rischio.

Il problema di fondo è che oggi esiste un gran numero di domande che non ha una risposta; in altre parole esistono problemi per i quali non esistono procedure ben definite che con un ragionevole numero di passi possono giungere direttamente dalla domanda alla risposta tramite calcoli. Esiste invece l'assurdo che, per un problema di difficile soluzione, non sempre il suo inverso sia altrettanto difficile; un esempio è dato dal ben noto problema di ricavare una funzione numerica dalla sua rappresentazione grafica; tanto per capirci, passare dalla rappresentazione numerica alla grafica è cosa di estrema facilità; è invece, nella stragrande maggioranza dei casi, impossibile umanamente recupere l'equazione numerica di una curva anche poco complessa dalla sua rappresentazione nel piano. La tecnica della domanda-risposta cui prima si accennava può essere applicata qui, facendo tracciare alla macchina una serie estremamente numerosa di curve in base a funzioni numeriche all'uopo ipotizzate e poi per confronto ricavare la funzione che più si approssima a quella del problema. Questo metodo, noto nell'area scientifica come della «funzione inversa», è utilizzato diffusamente in molte aree applicative (ingegneria in primis), dove esigenze di rapidità e ampie possibilità di approssimazione permettono di adottare metodi non proprio esatti in un numero elevato di problematiche.

È questo il metodo delle tabelle di Thompson, impiegato nelle partite a scacchi di cui discutemmo la volta scorsa. Purtroppo il grosso guaio di questa tecnica è che spesso ci si trova di fronte a risultati di cui non si sa spiegare la via per la quale ci si è arrivati.

COMPUTER HSP COMPUTER

COMPUTER HSP COMPUTER

COMPUTER HSP COMPUTER

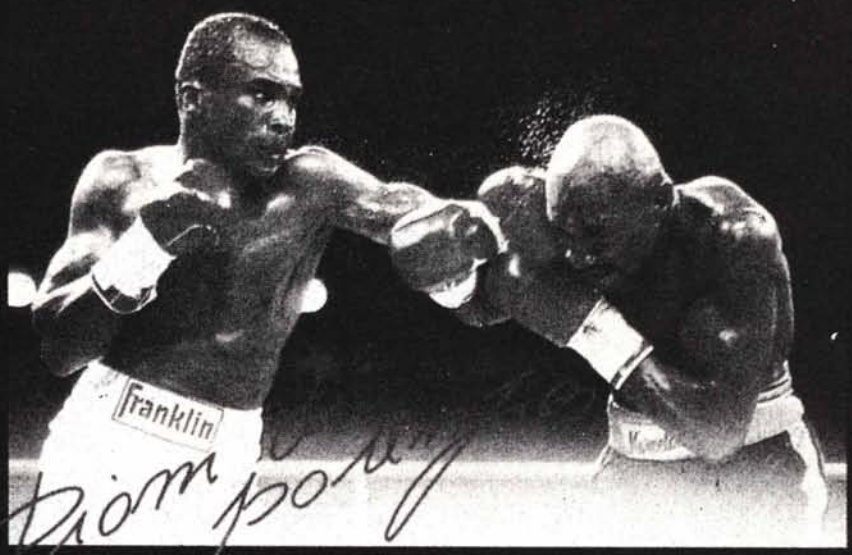
COMPUTER HSP COMPUTER

COMPUTER HSP COMPUTER

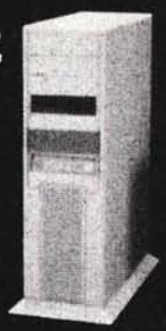
COMPUTER HSP COMPUTER

COMPUTER HSP COMPUTER

THE BIG APPLE



386 33Mhz CACHE
da £. 3.299.000



ANNO 1989

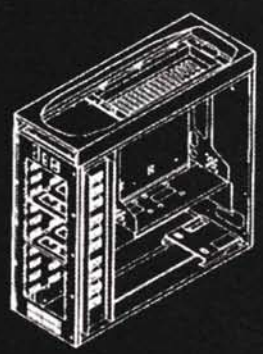
64K CACHE, 4MB RAM,CTRL 2HDD +2FDD,FDD 1 2MB

486 25Mhz CACHE
da £. 6.990.000

ANNO 1990

8K CACHE,4MB RAM,CTRL 2HDD+2FDD 1.1,FDD 1,2MB

.....



ANNO 1991

Uff. Comm.: Via P. Fumaroli 12/A Tel. 06 – 2251517 – ROMA
Conc. Centro Italia Info.Sist.: Via Malta 8 – Tel. 06–8842378/8411987 – 00198 ROMA
Centro ass. PC Service Via Malta 8 – Tel. 06 – 8411987 – 00100 ROMA