di Raffaello De Masi

Come è organizzato un sistema esperto?

Nella puntata scorsa abbiamo verificato come il vero cuore di un sistema esperto sia rappresentato dal suo blocco di conoscenza, strutturato in maniera opportuna per prendere decisioni. Come è organizzata questa base di conoscenza per rispondere in maniera efficiente a questa problematica?
La risposta non è semplice organizzare un sistema esperto non è come redigere un programma in cui lo sviluppo della problematica è lineare e i vincoli di libertà del problema sono ben noti a chi redige il programma; una base di conoscenza è paradossalmente tanto meno efficiente quanto più è vasta: l'assurdo si intende se si tiene conto che una base di dati ampia abbraccia aspetti sempre più estesi e interconnessi del problema

Un sistema esperto, come abbiamo visto la volta passata, è rappresentato da una messe, un fascio di regole e fatti, tra loro interconnessi, che manipolano la base di conoscenza; ma non è detto che tali fatti e regole siano sempre veri o falsi, nell'ambiente, e, soprattutto, cosa che complica all'ennesima potenza il ragionamento, non sempre sono interamente veri o falsi.

Capiremo meglio il tutto con un esempio.

Fatti:
- la tanica n. 23 contiene acido solforico
- l'utente è rimasto ferito nell'uso di una falciatrice

regole:
- se la ricerca dello ione solfato è positiva il materiale contenuto nella tanica è acido solforico
- se l'utente ha maneggiato in maniera negligente la falciatrice, è possibile applicare una ipotesi di concorso di colpa nella disgrazia.

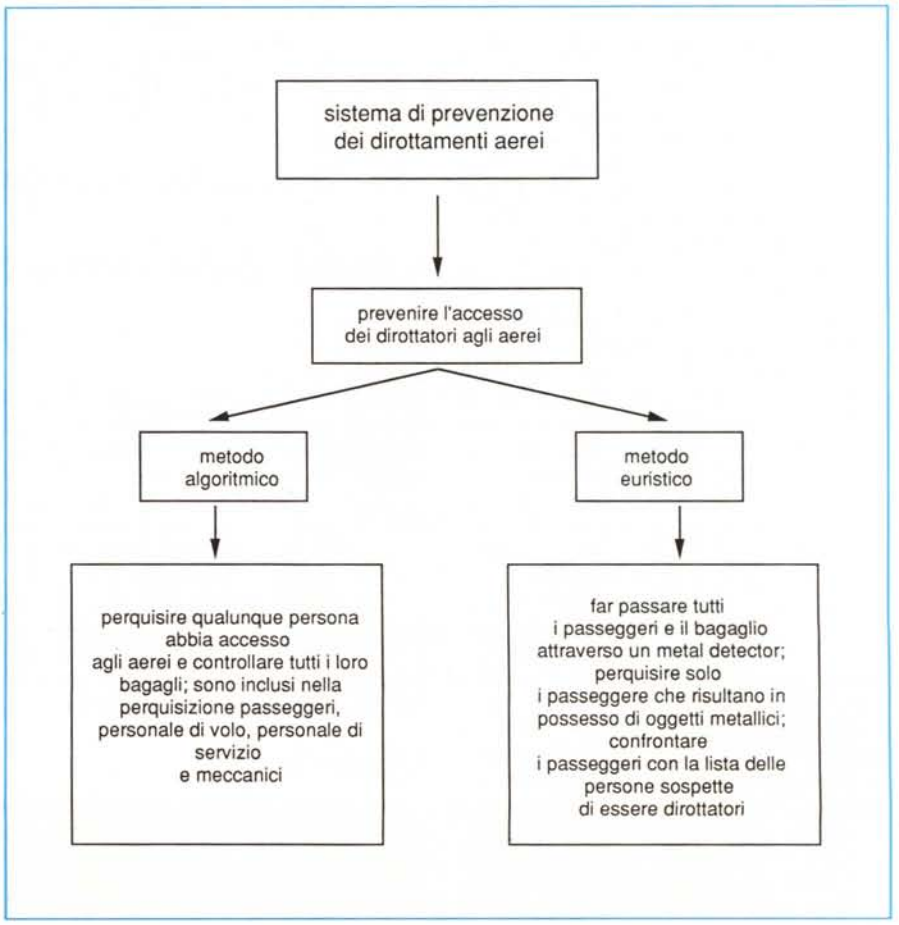

Figura a - Confronto tra la tecnica algoritmica e quella euristica.

Fatti e regole, in un sistema esperto, non sono, come dicevano sempre veri o falsi al 100%; il grado di certezza può essere inferiore all'assoluto; in questo caso si parla di «fattore di certezza». Il tutto è esemplificato di seguito:

fatti:
- il magazzino 55 contiene la tanica 35 che contiene alcool etilico con certezza 1.0
- la falciatrice è difettosa con certezza 0.8

regole:
- se il materiale fuoriuscito è acido solforico con certezza 1, il magazzino di provenienza del materiale perso è il 55, con certezza 0.9
- se la falciatrice era difettosa con certezza superiore a 0.5, la teoria della inaffidabilità del prodotto è sicuramente di 1.

Come si vede non esiste una stretta correlazione numerica tra le diverse parti della regola; in base a che cosa, ad esempio, viene stabilita la percentuale di certezza, visto che, almeno nelle regole sopracitate, non esiste tecnica per la determinazione, il «dosaggio» di tale percentuale?

La risposta farà storcere il naso ai puristi del calcolo matematico; ma tant'è. Molte delle regole di un sistema esperto non hanno piena e totale corrispondenza in una vera e propria analisi matematica del problema. In gergo si dice semplicemente che molte regole sono *«euristiche»*, regole dettate dalla pratica e dall'esperienza che semplificano anche notevolmente il problema e limitano efficacemente lo sforzo di ricerca della soluzione. Tutti i sistemi esperti usano tecniche euristiche, specie nel caso di problematiche in cui il grado di incertezza è tale che una corretta gestione parametrica del problema sarebbe estremamente difficoltosa e tipicamente poco sicura. Si tratta, come dicevamo, di qualcosa che cozza in maniera anche feroce con le normali tecniche di soluzione tipiche di altri linguaggi e sistemi, basati su una soluzione algoritmica del problema. Detto in altre parole,

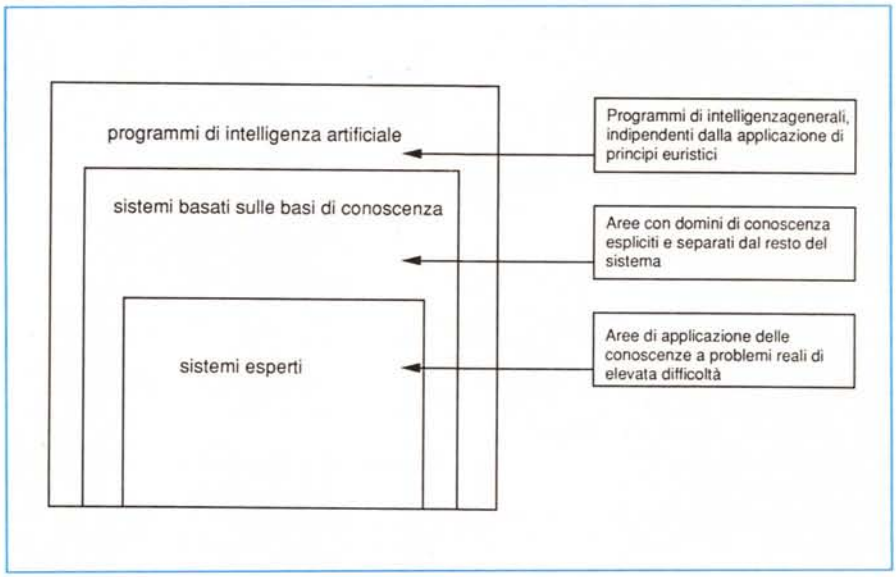

Figura b - La posizione gerarchica dei sistemi esperti nei programmi di Intelligenza Artificiale.

un metodo algoritmico garantisce la corretta od ottimale soluzione al problema, mentre un metodo euristico produce una soluzione accettabile nella maggior parte dei casi, ma semplificando in maniera essenziale la vita all'uomo e alla macchina.

Nella figura a si vede la differenza fra metodi di soluzione attraverso un algoritmo o una ricerca euristica. La figura si commenta da sé e, a parte l'effettiva applicazione ad un fatto quotidiano, simula quanto effettivamente, nella vita normale, viene applicato nella soluzione dei diversi problemi. In altri termini ambedue i provvedimenti espressi nella figura portano alla stessa soluzione; il primo dà certezza assoluta, il secondo una *ragionevole* certezza.

La tecnica algoritmica fornisce, nell'esempio citato, certamente la soluzione ottimale per l'eliminazione dei dirottamenti aerei, in quanto esclude virtualmente qualsiasi possibilità che qualcuno possa salire a bordo di un aereo munito di un'arma. Purtroppo ha il difetto di avere tempi di realizzazione lunghi, di essere costoso in termini di mezzi e di danaro, e tanto impopolare da perdere qualunque valore pratico. La tecnica euristica, pur non garantendo iL 100% dei risultati, rende comunque la ricerca delle soluzioni molto più facile e pratica.

Come abbiamo già diverse volte avuto modo di far notare, la conoscenza, in un sistema esperto, è organizzata in modo da essere completamente separata dal dominio delle regole. La struttura fondamentale può essere rappresentata, tout court, da due blocchi separati e fondamentali; la collezione delle conoscenze detta anche «base di conoscenza», e le regole per il maneggio di tale base, detto «motore inferenziale». Un programma con conoscenza organizzata in tal modo è chiamato sistema basato sulla conoscenza.

La figura b mostra i rapporti tra l'intelligenza artificiale, i sistemi basati sulle basi di conoscenza e i sistemi esperti. In base ad essa si evince come tutti i sistemi esperti siano basati su sistemi

di basi di conoscenza, ma che non è vero il contrario. Ancora, non tutti i programmi dotati di motore inferenziale e di una base di conoscenza sono sistemi esperti.

La base di conoscenza in un S.E., cosa questa che è già evidente dall'esame di quanto precedentemente esposto, contiene fatti (dati) regole o rappresentazioni di esse; le regole, come abbiamo già visto, usano questi fatti sulla base delle determinazioni euristiche che consentono di giungere a una decisione ragionevolmente esatta. Il motore inferenziale contiene invece un interprete che decide quale regola applicare per poter aumentare la base di conoscenza e uno scheduler che organizza le regole da sviluppare e il loro ordine di esecuzione. L'organizzazione schematica del tutto è riportata nella figura c.

Il grandissimo vantaggio di tale organizzazione dell'ambiente è dato soprattutto dal fatto che regole e conoscenza sono separati tra loro. In questo modo è più semplice, per l'ingegnere della conoscenza intervenire nella manipolazione delle une e dell'altra. La cosa di basilare importanza è stabilire una corretta procedura di utilizzazione della base di dati, in quanto un sistema esperto è davvero utile ed efficace se e soltanto se possiede conoscenze abbastanza vaste e appropriate e ancora i mezzi per usare questa stessa conoscenza in maniera efficace tanto da giungere a risultati rapidi e concreti. Detto in altre parole, un sistema esperto, per essere efficace, deve avere una base di conoscenza sufficientemente aggiornata e potente nei confronti del problema, e un motore inferenziale ben costruito e adeguatamente tagliato al maneggio di tale base.

Il concetto di motore inferenziale non è sempre immediato e ben capito da chi si avvia allo studio delle discipline di Intelligenza Artificiale. In genere è abbastanza chiaro e intuitivo, qualunque linguaggio si utilizzi, come manipolare la base di dati e scrivere i fatti e le regole destinate a gestirla; è invece meno chiaro vedere come tutte le regole possano essere organizzate in un motore inferenziale. Tutto ciò generalmente avviene in quanto non esiste alcun metodo, semplice e generale, per costruire un motore inferenziale (che da questo momento, per evitare ripetizioni cacofoniche, chiameremo talvolta semplicemente «motore», tout court). Come un motore è organizzato e strutturato dipende sia dalla natura stessa del problema, sia dal modo con cui si desidera strutturare e costruire il motore stesso. A ciò si aggiunge spesso qualche complicazione da parte dei costruttori di linguaggi. Alcuni di essi, anche ad alto livello, come EMYCIN (van Melle W, Shortliffe E.H., e Buchanam B.G. - «EMYCIN, a domain-indipendent system aiding in construction of knowledge-based consultation programs». In Machine Intelligence, Infothec State of Art Report, serie 9, n. 3, Pergamon Info. Ltd, poi pubblicato come volume da Pergamon Press, 1981) possiedono il motore inferenziale direttamente implementato nel linguaggio, come sua parte. Altri, forse di concezione più antiquata, abbisognano di una parte dell'ambiente, generalmente separata dal linguaggio principale, che costruisce, generalmente a posteriori, il motore.

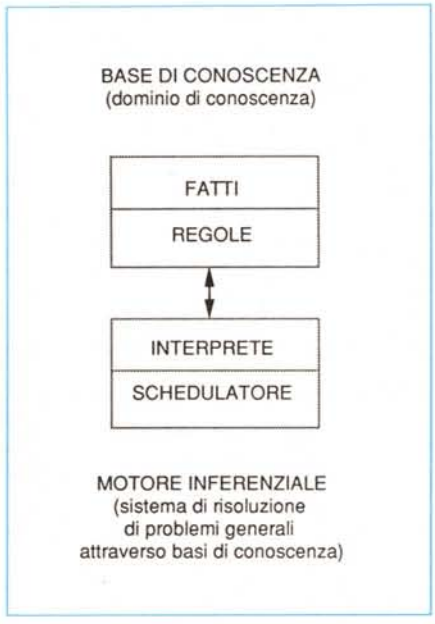

Figura c - La struttura di un sistema esperto.

Ambedue gli approcci, come autorevolmente riferisce D. Waterman in «Building ecc.» opera già più volte nominata su queste pagine e da cui i nostri scritti derivano spunti e citazioni in gran numero, hanno i loro svantaggi e vantaggi. Nel primo caso, un motore inferenziale direttamente costruito dal linguaggio sovente in maniera del tutto trasparente risparmia gran lavoro al programmatore. Al contrario, cosa altrettanto intuibile, nel secondo caso il programmatore ha maggiore capacità di controllo sulla costruzione del motore e non è soggetto a subire qualcosa di prefabbricato che potrebbe non essere corrispondente o adeguato a quanto da lui richiesto o alla base di conoscenza costruita. Un linguaggio a basso livello (come ad esempio un LISP prima maniera) senza motore del tipo built-in potrà dare maggiori difficoltà al programmatore, ma fornisce una possibilità di accesso più curato e particolare.

Come al solito «in medio stat virtus», vale a dire che linguaggi ancora più nuovi, come il FERO-L hanno la possibilità di escludere o meno la creazione automatica del motore; i vantaggi sono evidenti specie nel passaggio successivo a gradi di esperienza sempre maggiore del programmatore; sovente il motore built-in viene settato all'inizio, mentre poi, in fase di debug e di affinamento si può procedere, come dicono gli addetti ai lavori, a strutture più self-tailored.

Un esempio del genere è rappresentato da un ambiente per la costruzione di sistemi esperti, HEARSAY-III, che possiede una serie di tool, routine precostruite, destinati alla realizzazione del motore, cui il programmatore può accedere per ridefinirne o potenziarne parti o meccanismi. C'è però da dire che, dal 1985 in poi, con la comparsa di CARAT/PM, capostipite di una nutrita serie, i linguaggi di alto livello producono motori inferenziali di grandissima efficienza, attraverso un abile pilotaggio, da parte del programmatore, delle esigenze espresse in default o durante la stesura delle regole e dei fatti stesi.

La rappresentazione della conoscenza

Infine consideriamo come la conoscenza è organizzata in una «rappre-

sentazione strutturata» della conoscenza stessa. Esiste una serie di tecniche abbastanza standard di rappresentazione della conoscenza, che possono essere utilizzate, singolarmente o in gruppo, per la costruzione dei sistemi esperti. Anche qui ogni tecnica porta al costrutto totale i suoi benefici, come renderlo più veloce, più efficiente (si ricordi che le due cose, almeno in AI non sono sinonimi), più leggibile, più facilmente aggiornabile e manutenibile, ecc. Parleremo probabilmente abbastanza presto di queste tecniche. Si ricordi solo, almeno per adesso, che le più utilizzate si basano sulla redazione di regole (la tecnica più usata, specie dai principianti, e in base alla quale è prodotto circa il 50% del materiale presente sul mercato), sulle reti semantiche, e sui blocchi (i testi specializzati non sono d'accordo su questi termini, per cui ve ne forniamo le definizioni originarie: rules, semantic net e frame).

I metodi basati sulla redazione delle regole

I metodi basati su questa tecnica sono i più facili, come dicevamo, da apprendere. Il motore inferenziale è rappresentato in parole molto povere da una serie di costrutti condizionali del tipo IF-THEN. Ad esempio:

- IF il signor Tizio lavora in un reparto di verniciatura

THEN ha avuto con grande probabilità occasione di manipolare solventi.

- IF il signor Tizio ha maneggiato per lungo tempo solventi

AND ha lavorato in spazi chiusi senza protezione

THEN il paziente corre gravi rischi di affezione cancerosa.

Secondo una tecnica vecchia quanto i calcolatori, se la situazione di IF soddisfa alla regola, tutto quanto dopo il THEN viene verificato e eventualmente eseguito. È importante notare che le conclusioni possono coinvolgere il mondo esterno (ad esempio stampa di un messaggio sul video o sulla stampante), eseguire una ulteriore serie di controlli, o istruire il sistema a leggere una conclusione.

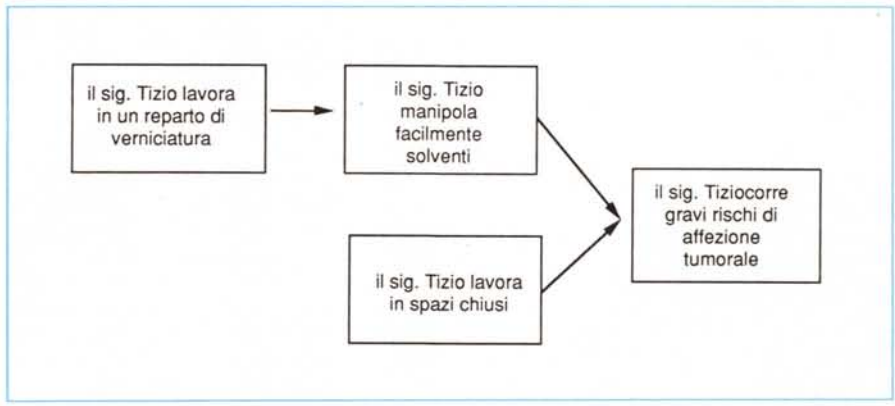

Figura d - Un esempio di semplice catena inferenziale nella diagnosi di una malattia.

L'esecuzione di quanto supposto dall'IF iniziale (che ovviamente può avere una struttura composita, ben oltre quanto ci avevano abituato a vedere linguaggi all-purpose come Fortran o C) può produrre quelle che sono anche definite «inference chain». Si forma cioè una catena inferenziale che, tenendo conto di quanto esposto nell'esempio, è rappresentata nella figura d; nel caso particolare la catena mostra le componenti che hanno portato alla diagnosi tumorale; è ovvio che una catena può essere, in un sistema esperto sufficientemente sviluppato, notevolmente articolata e complessa.

La tecnica delle regole fornisce una via naturale e semplice nella redazione e nella costruzione di un motore; in termini più precisi è possibile affermare che scopo delle regole è condurre il programma in modo tale da reagire al cambiamento dei dati in input senza richiedere una conoscenza avanzata circa il flusso di controllo. In un programma convenzionale il flusso di controllo e l'uso dei dati sono predeterminati dal programma stesso; come avemmo già modo di far notare qualche puntata addietro quando parlammo della differenza tra i linguaggi di AI e quelli più convenzionali.

In questi (e in quelli di AI basati su regole), lo sviluppo avviene per passi successivi, e salti o decisioni avvengono solo in punti predeterminati. Ciò funziona alla perfezione con problemi solubili in maniera algoritmica, in cui i dati cambiano in maniera relativamente lenta, ma le cose peggiorano rapidamente quando si lavora con grandi messi di dati, dove i salti e le decisioni sono la norma invece dell'eccezione. Inoltre un programma redatto con la tecnica delle regole riesce più facilmente leggibile e documentabile.

Gli altri due metodi principali descritti (reti semantiche e blocchi) possiedono una struttura simile, tanto da poter essere descritti, almeno adesso, insieme. La struttura è sottilmente simile a quella precedente. La conoscenza (anzi la sua rappresentazione) è organizzata in un blocco di nodi connessi da relazioni e organizzati in gerarchia. Ogni nodo rappresenta concetti che sono esplicati da attributi e valori connessi col nodo stesso. I nodi a livello più basso nella gerarchia (più ramificati) ovviamente «specializzano» le proprietà di quelli più alti.

Si tratta anche qui di un concetto abbastanza intuibile, ma che è più difficile da implementare in maniera efficiente da una persona non esperta. Ma di tutto ciò parleremo a lungo prossimamente, quando passeremo alla costruzione di un vero sistema esperto.

Uno dei più grossi handicap per chi si avvicina ad un sistema esperto è rappresentato dal fatto che già ha programmato in qualche altro linguaggio; purtroppo gli idiomi di AI sono tanto diversi da quelli più convenzionali che riesce paradossalmente meglio chi, di linguaggi di programmazione non ha mai sentito. Perciò la prossima puntata cercheremo di chiarire in che cosa il linguaggio di AI e, in particolare i sistemi esperti differiscono dai linguaggi e dai programmi (rispettivamente) tradizionali. MC