Honeywell
Honeywell Information Systems Italia
Via Laboratori Olivetti - Pregnana Milanese

UNIVERSITÀ
DEGLI STUDI DI MILANO
DIPARTIMENTO DI SCIENZE
DELL'INFORMAZIONE
Via G. Colombo, 49
20133 Milano - Tel. 2772234

35

NOTE DI SOFTWARE

UNIVERSITÀ
DEGLI STUDI DI MILANO
DIPARTIMENTO DI SCIENZE
DELL'INFORMAZIONE

Honeywell
Honeywell Information Systems Italia

Note di software.
è un bollettino trimestrale, edito a cura del Centro di Ricerca e Progettazione della Honeywell Information Systems Italia e del Dipartimento di Scienze dell'Informazione dell'Università di Milano.

Note di software.
intende costituire uno strumento per la rapida e informale circolazione di informazioni, idee ed esperienze nell'area della tecnologia del software. In particolare, intende stimolare il dibattito sulle metodologie orientate alla diminuzione del rapporto costo/qualità dei programmi, e sugli strumenti atti ad automatizzare l'attività di produzione del software in ogni suo aspetto.

La collaborazione a **Note di software** è libera.
In particolare, si sollecitano contributi nella forma di brevi monografie e bibliografie essenziali commentate sui seguenti temi: tecniche e strumenti per la programmazione strutturata e modulare, metodologie e strumenti per il testing e il debugging dei programmi, documentazione del software, aspetti organizzativi e gestionali nella costruzione di sistemi software di grandi dimensioni.

Chi desiderasse pubblicare un contributo è pregato di prendere contatto con il Comitato di Redazione (HISI - Via ai Laboratori Olivetti - Pregnana Milanese), o di inviare direttamente il dattiloscritto in triplice copia.
I contributi non dovrebbero superare le quattromila parole.

I lavori accettati per la pubblicazione vengono revisionati dal Comitato di Redazione, che si avvale del contributo di specialisti del settore. Ogni contributo pubblicato riflette, comunque, le opinioni dei suoi autori, e non necessariamente quelle del Comitato di Redazione. Eventuali revisioni saranno effettuate di comune accordo fra il Comitato di Redazione e i proponenti.

Note di software.
Viene distribuito ad aziende, Enti e specialisti del settore che ne facciano richiesta alla redazione.

Direttore responsabile: Paolo Ghelfi
Comitato di Redazione: Roberto Polillo - Wanda Anselmetti - Stefania Bandini - Francesco Gardin
Redazione: Franco Soresini

35

Marzo 1987

Indice:

Autorizzazione del tribunale di Milano n. 87 del 23 febbraio 1977

NOTE DI SOFTWARE

QUESTO NUMERO

Questo numero di NOTE DI SOFTWARE, interamente dedicato all'Intelligenza Artificiale, si apre con un lavoro di Domenico Parisi in cui viene presentato il nuovo paradigma del connessionismo, che coinvolgendo tre discipline quali l'Intelligenza Artificiale, la Psicologia e le Neuroscenze, propone una nuova chiave di lettura dell'Intelligenza Artificiale stessa in cui l'intelligenza e l'apprendimento sono rappresentate in nodi e nelle loro connessioni. L'autore, oltre ad una panoramica sul connessionismo ed il contesto tecnologico-culturale nell'ambito del quale si inserisce, propone alcuni esempi relativi all'interpretazione del linguaggio naturale in cui alcune delle proprietà dell'approccio connessionista vengono particolarmente evidenziate.

Il secondo articolo, di P. Dall'Orco e R. Giorgesi, analizza alcuni lavori nell'ambito della rappresentazione della conoscenza temporale, classificandoli in una serie di modelli. Per ognuno di essi vengono presentate le caratteristiche più significative e le classi di problemi per i quali sono più adeguati. Si tratta indubbiamente di una "survey" molto interessante dato l'interesse che tale tema sta assumendo nella comunità dell'Intelligenza Artificiale in senso lato e tra coloro che intendono realizzare sistemi intelligenti dove il fattore temporale deve essere trattato adeguatamente.

Il lavoro di Adriana Batistoni, Mauro di Manzo, Franca Ricci e Emanuele Trucco affronta un problema centrale nel settore della visione artificiale: l'individuazione della conoscenza necessaria per guidare il processo di riconoscimento. L'utilizzazione di conoscenza funzionale relativa agli oggetti, oltre che a quella geometrica, opportunamente integrata con questa, fornisce un modello di descrittori semantici basato su primitive funzionali. Tale forma di rappresentazione, oltre che al riconoscimento dell'oggetto, rappresenta la base per poter ragionare relativamente alle sue funzionalità.

L'articolo di Paolo Stofella e Hadley Taylor, descrive la realizzazione di un sistema per la rappresentazione e l'uso della conoscenza relativa alla fisica naive, ossia una fisica basata su leggi intuitive di comportamento del mondo fisico. Il tipo di fenomeni trattati dal modello presentato è relativo ai liquidi. Il simulatore sviluppato è inoltre integrato con un interprete che analizza l'evoluzione del processo simulato e lo descrive in termini di primitive ad alto livello. L'integrazione di questi due componenti con un modulo di ragionamento rappresenta un interessante sistema di problem solving.

Il lavoro di Stefania Bandini e Fabio Buciol descrive un sistema basato sul formalismo di rappresentazione della conoscenza a frame, realizzato in Prolog. Viene inoltre illustrata la sintassi mediante la quale è possibile definire contesti. I frame sono gestiti globalmente mediante una rete semantica che oltre a collegarli ne permette la gestione dinamica.

L'articolo di Emilio Bertolotti presenta un'interessante progetto di basi di conoscenza finalizzate al monitoraggio in tempo reale nel controllo di processo. Si tratta di un sistema esperto realizzato presso l'Honeywell Computer Science Center di Minneapolis che riguarda il problema della produzione di cellulosa e il monitoraggio degli impianti ad essa dedicati.

Sempre nel campo dei problemi delle applicazioni di sistemi esperti per il controllo in tempo reale di processi industriali Alessandro Mazzetti presenta un lavoro in cui viene descritto un ambiente di sviluppo di basi della conoscenza realizzato nell'ambito del programma inglese di ricerca Alvey.

Infine, nella rubrica dedicata alle recensioni, Emanuele Ciapessoni presenta un commento al libro curato da D. DeGroot e G. Lindstrom "Logic Programming, Functions, Relations and Equations".

La REDAZIONE

CONNESSIONISMO: UN NUOVO PARADIGMA NELLO STUDIO DELLA MENTE (*)

Domenico Parisi

Reparto Processi Cognitivi e Intelligenza Artificiale

Istituto di Psicologia - C.N.R.

Roma

RIASSUNTO

I sistemi connessionisti, o con parallelismo elevato, rappresentano una svolta concettuale e tecnologica che può avere conseguenze importanti in Intelligenza Artificiale. Il passaggio dalle tradizionali architetture sequenziali ad architetture in cui l'intelligenza del sistema risiede nel pattern di interconnessioni tra nodi di una rete e nella attivazione che si propaga in parallelo nella rete, può consentire di riprodurre aspetti finora necessariamente trascurati dell'intelligenza quali la flessibilità, la sensibilità al contesto e l'apprendimento spontaneo con l'esperienza.

ABSTRACT

Connectionist, or massivly parallel systems, represent a conceptual and technological breacktrough which can have important consequences in Artificial Intelligence. Moving from traditional sequential architectures to architectures where the system's intelligence resides in the pattern of connections among nodes of the network and the parallel propagation of activations, allows the implementation of intelligence's features so far such as flexibility, context sensitivity and spontaneous learning with experience.

1. INTRODUZIONE

Un nuovo paradigma sta emergendo nello studio della mente e dell'intelligenza: il connessionismo, ovvero i modelli a parallelismo elevato (massively parallel systems). Queste espressioni si riferiscono al fatto che il funzionamento mentale e l'attività intelligente vengono concepiti fondamentalmente come attivazione che si propaga in parallelo nella rete, subisce aggiustamenti dinamici a causa delle interazioni tra le varie parti della rete, e poi si assesta in un equilibrio ottimale. Il paradigma connessionista si va sviluppando contemporaneamente all'interno di tre discipline che si occupano in un modo o nell'altro della mente: l'intelligenza artificiale, la psicologia e le neuroscienze. In effetti, proprio per questa ragione il connessionismo appare come la prima possibilità di dare un contenuto effettivo e una base sostanziale e unificante al concetto di scienza cognitiva, che finora non è stata molto più che l'idea di una collaborazione tra discipline diverse, con un vago richiamo all'elaborazione dell'informazione e ai calcolatori elettronici come tratto comune.

2. ARCHITETTURE SEQUENZIALI E ARCHITETTURE PARALLELE

All'interno dell'informatica e dell'intelligenza artificiale il connessionismo è emerso di recente come una questione tecnologica, cioè come il tentativo di costruire sistemi computazionali a paralellismo elevato. L'architettura classica dei calcolatori elettronici, definita 30 anni fa da John von Neumann [16], è una architettura sequenziale. Il sistema è costituito fondamentalmente da due componenti: il processore centrale (la CPU, central processing unit) e la memoria. Il processore centrale esegue una sequenza di istruzioni (il programma) sulle informazioni conservate in un gran numero di unità memoria. La memoria è passiva ed è per la maggior parte del tempo inutilizzata, in quanto sono di volta in volta in azione solo le unità sulle quali il processore sta eseguendo le istruzioni del programma. Il problema tecnologico di questa archi-

(*) Questo articolo verrà pubblicato dal Giornale Italiano di Psicologia.

tettura - che è del resto quella di quasi tutti i calcolatori, grandi o piccoli, oggi in uso - è che tutto il sistema dipende dalla interazione tra il processore e la memoria e questa interazione - cioè l'esecuzione sequenziale delle istruzioni del programma una alla volta sulla memoria - costituisce un "collo di bottiglia" che rallenta tutto il processo e compromette l'efficienza del sistema. La via d'uscita che è stata tentata è stata ovviamente quella di aumentare la velocità di esecuzione delle singole istruzioni del programma da parte del processore: circa un milione di istruzioni al secondo per un personal computer, da 30 a 40 milioni di istruzioni al secondo per un calcolatore di grandi dimensioni. Ma questa strada incontra dei limiti, anche fisici, e comunque non sembra capace di portare la velocità e quindi l'efficienza del sistema al di là di certi livelli. Questa appare una seria limitazione dell'architettura sequenziale classica di von Neumann, che si fa sentire da tempo nell'uso dei calcolatori per il calcolo scientifico (ad esempio, in fisica) ma che ormai comincia ad essere evidente anche nell'uso dei calcolatori per la riproduzione delle attività mentali, simboliche, intelligenti, da quelle più elementari della percezione e del movimento a quelle più complesse del ragionamento, dell'inferenza, della comprensione del linguaggio, della soluzione dei problemi.

In effetti, da qualche tempo si compiono sforzi per uscire dal collo di bottiglia delle architetture sequenziali introducendo elementi di parallelismo nei sistemi. Ad esempio in alcuni sistemi l'esecuzione di una istruzione comincia quando ancora non è terminata l'esecuzione dell'istruzione precedente, consentendo così un certo grado di parallelismo. In altri sistemi, esistono diversi processori che si dividono il compito di eseguire parti diverse del programma. Tuttavia, in queste versioni di parallelismo limitato rimane ancora sotanzialmente immutata l'idea di uno o più processori che eseguono le istruzioni di un programma e di una memoria passiva su cui le istruzioni vengono eseguite. I sistemi a parallelismo elevato (massively parallel systems) sono invece un tentativo di uscire in modo radicale dalla concezione sequenziale tradizionale dell'architettura dei calcolatori. In questi sistemi - ad esempio nella Connection Machine di Daniel Hillis [4] -la distinzione tra processore attivo e memoria passiva si riduce sensibilmente, e l'accento per quanto riguarda il funzionamento e la stessa "Intelligenza" del sistema si sposta dal processore alla memoria. Quest'ultima viene concepita come una rete interconnessa di un gran numero (65.000 circa nella Connection Machine) di processori-memorie che si scambiano messaggi in parallelo, e le prestazione del sistema è il pattern di messaggi che vengono scambiati e l'assetto finale risultante dalle rete-memoria.

Da un punto di vista strettamente tecnologico un tale sistema a parallelismo elevato ha il vantaggio di utilizzare costantemente tutta la memoria - invece di lasciarne inutilizzata la maggior parte nella maggior parte del tempo, come avviene nei sistemi sequenziali - e soprattutto di raggiungere velocità di funzionamento molto più elevate (circa un milione di istruzioni al secondo nella Connection Machine). Non c'è quindi da sorprendersi se la costruzione di macchine parallele e lo sviluppo di software capace di girare su queste macchine siano oggi uno dei settori di punta della ricerca in informatica e in intelligenza artificiale, dal progetto giapponese dei calcolatori della "quinta generazione", ai diversi progetti pubblici e dell'industria privata negli Stati Uniti (la Connection Machine ha cominciato a essere commercializzata quest'anno), fino al progetto italiano APE dell'Istituto Nazionale di Fisica Nucleare e al nuovo progetto finalizzato del Consiglio Nazionale delle Ricerche sul calcolo parallelo.

Ma il connessionismo e il parallelismo elevato non sono soltanto importanti sviluppi tecnologici. Essi sono soprattutto nuovi modi di pensare, cioè strumenti concettuali e metodologici di carattere generale per porsi problemi e disegnare soluzioni in modo nuovo rispetto al passato. Da questo punto di vista essi rappresentano un vero e proprio nuovo paradigma, in un senso non lontano da quello indicato da Kuhn [7] nella sua analisi del progresso scientifico. In questo lavoro cercheremo di illustrare alcune delle novità che il nuovo paradigma connessionistico introduce in una serie di campi in cui si studia la natura della mente e dell'intelligenza.

3. COMPUTAZIONE E CERVELLO

Il connessionismo e i modelli a parallelismo elevato stanno consentendo di porre in modo nuovo il problema dei rapporti tra lo studio astratto delle funzioni mentali e dell'intelligenza e lo studio del cervello come organo fisico.

I rapporti tra lo studio astratto e formale delle funzioni mentali (informatica e intelligenza artificiale) e quello concreto del cervello come macchina fisica (neuroscienze) ha una storia di più di 30 anni con tre fasi abbastanza ben distinguibili al suo interno. Nella fase iniziale, dal dopoguerra fino agli anni 60, questi rapporti sono stati abbastanza stretti. Vi era allora l'idea che fosse possibile studiare con metodi computazionali sia la mente che il cervello, e che i due studi potevano essere reciprocamente utili. Questo appare nei titoli dei libri dei principali autori dell'epoca: CIBERNETICS: COMMUNICATIONS AND CONTROL IN MACHINES, ANIMALS AND MEN di Wiener, THE COMPUTER AND THE BRAIN di Von Neumann, EMBODIMENTS OF MIND di McCulloch, PRINCIPLES OF NEURODYNAMICS di Rosenblatt. L'espressione "cibernetica" indicava questo studio integrato della mente e del cervello, a cui contribuivano informatici, matematici e neurofisiologi e che hanno prodotto i modelli formali delle reti neuroniche e i sistemi di apprendimento basati sul principio della auto-organizzazione.

Questa fase iniziale - e con essa l'uso dell'espressione stessa di cibernetica - in sostanza si esaurì all'inizio degli anni 60. Qualche risultato era stato raggiunto ma i tempi apparivano immaturi per uno sviluppo di "scienza normale" di questi studi. La valutazione che si affermò all'interno della comunità informatica nei riguardi delle cibernetica fu quella di un fallimento. L'intelligenza cominciò ad emergere - e con essa il termine stesso di "intelligenza artificiale", a scapito di quello di "cibernetica" - proprio

con l'autonomizzazione dello studio computazionale dell'intelligenza rispetto allo studio del cervello e fu completamente abbandonata l'idea che il cervello potesse essere una fonte di ispirazione per costruire modelli della mente o che i sistemi computazionali intelligenti dovessero confrontarsi con le evidenze empiriche sulle caratteristiche fisiche del cervello. D'altro canto l'intelligenza artificiale andava definendo con considerevole successo i suoi problemi (ricerca dell'informazione, soluzione dei problemi, pianificazione, inferenza, rappresentazione della conoscenza), e le sue aree di applicazione, i suoi strumenti concettuali e metodologici, e i suoi supporti tecnologici (calcolatori, linguaggi e ambienti di programmazione).

L'emergere del connessionismo e dei sistemi a parallelismo elevato alla fine degli anni '70 inaugura una terza fase nei rapporti tra studio informatico della mente e lo studio del cervello, una fase che per certi aspetti rappresenta un ritorno agli obiettivi iniziali riguardanti la possibilità di uno studio computazionale del cervello come macchina fisica e la utilità reciproca di un rapporto più stretto tra intelligenza artificiale e neuroscienze. Ovviamente oggi le cose sono parecchio cambiate rispetto a trent'anni fa. Ci sono stati sviluppi scientifici e tecnologici che autorizzano la previsione di maggiori successi in questo riaccostamento. Ma prima di vedere quali sono questi sviluppi cerchiamo di chiarire brevemente perchè il connessionismo e il parallelismo elevato sono oggi i mediatori di questo nuovo interesse reciproco tra intelligenza artificiale e neuroscienze.

Nonostante il titolo del libro di von Neumann, che stabilisce un collegamento tra calcolatore e cervello, l'architettura sequenziale di von Neumann non è un buon modello del cervello. Il cervello come macchina fisica sembra organizzato su principi diversi da quelli dei correnti calcolatori. Esso appare costituito da una rete formata da moltissimi neuroni (sull'ordine delle decine di miliardi) uniti tra loro da un numero elevato di interconnessioni (sull'ordine delle migliaia per neurone). L'operazione base - il "firing" di un singolo neurone - è un'operazione lenta, che prende un tempo dell'ordine di grandezza di millesecondi (quindi un massimo di mille operazioni per secondo).

La velocità e l'efficienza del sistema vengono ottenute attraverso il "firing" in parallelo di molti neuroni. Inoltre, le prestazioni del cervello - la sua "intelligenza" - appaiono derivare più dal funzionamento globale della rete di neuroni (il quadro complessivo di interconnessioni e attivazioni) che dalla esistenza di un meccanismo unico e localizzato di supervisione e controllo dei processi in atto. Come si vede, su un piano generale i sistemi connessionistici e a parallelismo elevato sono a prima vista molto più vicini a queste caratteristiche del cervello di quanto non siano i sistemi sequenziali classici. A questo si deve aggiungere che il comportamento del cervello appare possedere che non sono riprodotte con facilità e naturalezza in una architettura computazionale classica e, a dir la verità, neppure in molti dei sistemi costruiti finora in intelligenza artificiale. Tra queste proprietà vi sono le seguenti:

a) una grande flessibilità di prestazione dovuta alla capacità del processo in atto di tener conto del contesto, cioè di altri elementi in qualche modo presenti nella situazione ma che di per sé non fanno parte del processo in atto;

b) una capacità elevata di apprendimento, sviluppo, modificabilità per cui ogni attività del sistema lascia una traccia opportunamente filtrata che influenza le attività successive del sistema;

c) la compresenza di funzionamenti di tipo digitale, discreto, discontinuo, e di funzionamenti di tipo analogica e continuo (il concetto di soglia, centrale nel meccanismo di "firing" dei neuroni, è basato su questa compresenza). Queste proprietà del cervello (e quindi della mente) non sono ben riprodotte nei sistemi computazionali classici (e oggi più in voga), mentre appaiono, come vedremo, più facilmente e naturalmente affrontabili nell'ambito degli approcci connesionistici e di parallelismo elevato.

Ma, da un punto di vista generale, perchè oggi si può pensare di affrontare con maggiori speranze di successo rispetto a 30 anni fa il problema certamente non facile di stabilire una connessione tra studio formale e computazionale della mente e studio del cervello come macchina fisica? La ragione è semplice. In questi 30 anni sono stati compiuti progressi in vari campi che rendono il problema più trattabile. Innanzitutto vi sono gli sviluppi tecnologici che stanno conducendo alla costruzione di macchine e sistemi di tipo connessionistico e parallelo i quali, come abbiamo visto, sono più vicini alla architettura del cervello. In secondo luogo, rispetto a 30 anni fa oggi abbiamo tutti gli sviluppi dell'intelligenza artificiale intervenuti nel frattempo, cioè raffinamento di idee e metodi e la considerevole esperienza accumulata nella costruzione di sistemi computazionali. Si consideri peraltro che all'interno della stessa intelligenza artificiale degli anni 60 e 70 vi sono stati filoni tendenzialmente o esplicitamente vicini al connessionismo e al parallelismo (ricordiamo i nomi di Quillian e Fahlman), che oggi, con la nuova tecnologia che comicia ad essere disponibile, possono cominciare a tradursi in sistemi implementabili.

In terzo luogo, vi sono gli sviluppi, anche tecnologici, delle neuroscienze che certamente in questi trenta anni non sono state ferme ma hanno rappresentato uno dei settori più in crescita nella ricerca scientifica. È vero che lo studio delle proprietà fisiche del cervello da parte delle neuroscienze è più facile da realizzare se si affrontano i fenomeni più analitici e locali rispetto a quelli più integrati e globali che tipicamente interessano l'intelligenza artificiale, ma ormai è evidente che tra i livelli - sia sul piano della macchina fisica del cervello che sul piano delle capacità mentali - vi è continuità e quindi possibilità e utilità di scambi reciproci di conoscenze e di concetti, piuttosto che discontinuità e quindi isolamento e disinteresse reciproco.

Infine, c'è da considerare il nuovo interesse dei fisici per il cervello, in particolare per la possibilità di applicare a

certi comportamenti del cervello come macchina fisica idee e metodi di analisi che hanno origine nelle teorie sugli stati della materia, nelle teorie sulla distribuzione dell'energia in sistemi fisici assimilabili alle reti neuroniche, ecc. Questo interesse dei fisici prende oggi la forma di modelli non solo matematici ma anche computazionali e quindi va a innestarsi sul piano concettuale ma anche su quello metodologico (simulazione su calcolatore) in un generale approccio computazionale allo studio del cervello.

In ogni caso, giustificato o meno che sia, i segni di un ravvicinamento tra intelligenza artificiale e neuroscienze sono evidenti. Daniel Hillis afferma esplicitamente che la sua Connection Machine si fonda su principi fisiologici di funzionamento del cervello. Le ricerche attuali sulla basi fisiche della memoria (ad esempio, l'utilizzazione del glucosio da parte dei neuroni interessati a processi di apprendimento, vedi [5]) si pongono il problema se i sistemi di memoria siano locali o distribuiti, cioè un problema che oggi divide anche gli informatici e gli psicologi connessionisti [3], [13]. Le stesse ricerche sul NGF, il fattore di crescita neurofisiologica, che sono state premiate quest'anno con un Premio Nobel, concepiscono un cervello con forti capacità di adattamento, di modificazione e di crescita, tutte capacità che tendono ad essere tipiche dei nuovi modelli computazionali connessionisti. D'altro canto i modelli di connessionismo distribuito di Rumelhart, McClelland, Hinton e Sejnowski e altri si rifanno esplicitamente alle caratteristiche del cervello (vedi [13], capitolo 4).

Al di là di questi segni, le ricerche che dichiaratamente si pongono il problema di uno studio computazionale del cervello sulla base di modelli connessionistici e a parallelismo elevato sono ormai numerose. Queste ricerche, prodotte da informatici, fisici e psicologi (si veda [1], [13] e la bibliografia citata in questi lavori), rappresentano oggi una possibilità concreta di conoscenza del cervello a partire dalle sue proprietà più integrate e globali e quindi di un approccio "dall'alto" che deve saldarsi con l'approccio "dal basso" che sembra essere quello più accessibile alle neuroscienze e alle metodologie tradizionali (biofisica, neurobiologia, metodi sperimentali, ecc.).

4. ALCUNE PROPRIETÀ DEI SISTEMI CONNESSIONISTICI

Ma il connessionismo e i sistemi a parallelismo elevato rappresentano un paradigma nuovo nello studio della mente e dell'intelligenza anche indipendentemente dalla possibilità che offrono di ravvicinare lo studio della mente allo studio del cervello. Questo paradigma appare destinato a rivoluzionare lo studio della mente da parte degli psicologi e degli studiosi di intelligenza artificiale, e in effetti tocca gli stessi fondamenti filosofici dell'approccio computazionale allo studio della mente. Su quest'ultimo punto non possiamo qui intrattenerci a lungo, ma vogliamo soltanto osservare che alcune delle critiche che filosofi come Herbert Dreyfus hanno rivolto all'intelligenza artificiale [2] appaiono molto meno motivate nei riguardi di un approccio connessionistico. Tra queste critiche vi sono quelle che i sistemi computazionali hanno prestazioni rigide e poco sensibili al contesto, non hanno "storia" e esperienza, cioè ripartono sempre da zero nell'affrontare una nuova prestazione senza conservare traccia del passato, e infine sono capaci solo di funzionamenti discreti, digitali, e non di funzionamenti gradati e continui. Ora, anche i sistemi costruiti sui principi dell'intelligenza artificiale tradizionale, cioè sequenziale e non connessionistica, possono in qualche modo rispondere a queste critiche, incorporando entro certi limiti le proprietà che li si accusa di non avere, ma è vero che questi sistemi non appaiono naturalmente concepiti e funzionanti per possedere queste proprietà.

Le cose stanno diversamente per i sistemi connessionistici e ad elevato parallelismo. Anche se non vi è spazio nel presente lavoro per una dimostrazione adeguata di questo punto, faremo riferimento alle nostre ricerche sulla comprensione del linguaggio naturale per indicare brevemente ma un po' più concretamente come si potrebbe dare una tale dimostrazione.

Nel nostro modello della comprensione del linguaggio ([10] in corso di stampa (a) e (b), [11]) il ruolo della conoscenza precedente (memoria) nel consentire al sistema di giungere ad una comprensione adeguata della frase o del testo che ha di fronte, viene concepito nel modo seguente. Le parole della frase in ingresso attivano ciascuna un nodo (o più nodi, se si tratta di parole che hanno più significati, es. *merlo*) nella rete che contiene e rappresenta la conoscenza del sistema. Ciascun nodo attivato cerca il percorso più breve che lo connette nella rete a ciascun altro nodo attivato. La conoscenza situata su questi percorsi minimi (PM) è quella che serve al sistema per risolvere problemi di comprensione (disambiguazione, comprensione del referente dei pronomi, inferenze, ecc.) e in sostanza rappresenta la comprensione che il sistema ha della frase.

Ora è possibile pensare che la ricerca del PM tra ciascuna coppia di nodi sia indipendente dalla ricerca del PM tra ciascuna altra coppia. Alternativamente si può invece assumere che non sia così, cioè che la ricerca del PM tra due nodi (o il PM già trovato tra due nodi) influenzi la ricerca del PM tra altri due nodi, o addirittura il PM che viene trovato. Questo nel nostro sistema comincia ad essere realizzato nel seguente modo. La ricerca del PM tra due nodi avviene con un passaggio in parallelo di attivazione da un nodo attivato a tutti i nodi che sono direttamente collegati a tale nodo (i suoi "vicini") e così via. (Con un'attivazione in parallelo, diversamente che con una attivazione in sequenza, la lunghezza del processo è solo funzione della distanza tra due nodi - non elevata - e non del numero medio dei vicini di un nodo - che può essere molto elevato). Il PM tra due nodi viene trovato appena il processo di attivazione che si propaga da un nodo incontra o l'altro nodo o il processo di attivazione che in parallelo si è propagato dall'altro nodo.

Ora, se assumiamo che un nodo resti attivato per qualche tempo e che il passaggio di attivazione da un nodo al suo

vicino sia funzione dello stato di attivazione del vicino da attivare, possiamo avere un modello in cui la ricerca del PM tra due nodi, o addirittura quale percorso viene trovato, non è un processo isolato dal resto, ma è potenzialmente influenzato da tutti gli altri processi di ricerca dei PM tra due nodi e, più in generale, dallo stato di attivazione dell'intera rete. Se si considera che i nodi da cui parte l'attivazione possono essere quelli delle parole della frase attualmente sotto processo ma anche i nodi attivati dal testo precedente o, al limite, i nodi attivati dalla situazione extralinguistica, percettiva, si ottiene un sistema in cui la maniera in cui una frase viene compresa appare fortemente sensibile al contesto, per quanto ampio, in cui la frase è inserita. D'altra parte un tale modello dell'influenza del contesto sul processo in atto e quindi sulla prestazione del sistema non appare limitato alla comprensione del linguaggio naturale ma può essere esteso a altre prestazioni intelligenti, se si assume, come noi assumiamo, che la rappresentazione della conoscenza e il modo d'uso di questa conoscenza (ricerca dei PM tra nodi) sia un meccanismo generale dell'intelligenza.

Al sistema così come lo abbiamo descritto non è difficile aggiungere ulteriore flessibilità e ulteriore somiglianza con i processi neuronici se si assume che:

a) l'attivazione di un nodo della rete non è un meccanismo a due stadi (attivo-non attivo) ma è meccanismo che ammette gradazioni, possibilmente continue;

b) anche le connessioni tra due nodi sono gradabili nel consentire il passaggio di attivazione;

c) esistono connessioni attivanti e connessioni inibenti l'attivazione.

Un sistema dotato di queste ulteriori sofisticazioni sarebbe non solo più flessibile ma potrebbe incorporare naturalmente e intrinsicamente alcune proprietà dei sistemi probabilistici e continui.

Infine se si allarga la nozione di contesto e di sensibilità al contesto che abbiamo introdotto più sopra così da ottenere che gli effetti delle prestazioni precedenti - cioè gli aggiustamenti dello stato della rete - non vengano riazzerati ma vengano mantenuti più o meno permanentemente, magari filtrati da processi di astrazione e da altri processi, e quindi possono permanentemente influenzare le nuove prestazioni del sistema, si ottengono sistemi che apprendono, si sviluppano, hanno "esperienza". Ad esempio, si può ottenere un sistema "esperto" in un determinato dominio (es. l'origine della pioggia) facendogli assorbire testi che parlano di quel dominio. Infatti, oltre a attivare nodi preesistenti, la comprensione di un testo porta con sé la creazione di nuovi nodi e di nuove connessioni tra nodi. In questo modo si può passare da un sistema che è un "novizio" in quel campo a un sistema che è un "esperto". (Sulla concezione della rappresentazione della conoscenza come una memoria dinamica, si veda [14] e [6]).

5. DETERMINISMO, NON-DETERMINISMO, CONNESSIONISMO

Il connessionismo consente di affrontare in modo nuovo e forse più risolutivo problemi fondamentali dell'intelligenza artificiale e della stessa informatica. Si consideri il problema che viene detto del determinismo-nondeterminismo dei sistemi computazionali. Noi esamineremo questo problema facendo riferimento a quell'area centrale dell'intelligenza artificiale che è la comprensione del linguaggio (natural language understanding), ma il problema ha evidentemente una portata più generale in intelligenza artificiale.

Una frase o un testo che debbono essere processate da una sistema automatico di comprensione costituiscono un compito con evidenti caratteristiche di sequenzialità. La frase deve essere processata un pezzo alla volta, diciamo una parola alla volta, da sinistra a destra considerando un testo scritto. L'ostacolo forse più grosso si incontra nel costruire un sistema automatico di comprensione è quello dell'ambiguità. Nel processare la frase sequenzialmente il sistema incontra continuamente punti in cui il modo giusto di procedere non è univoco, ma sono possibili varie strade di cui una soltanto è quella corretta. C'è una grande varietà di tipi di "ambiguità" nel processamento di una frase, dalla ambiguità di una parola come *merlo,* che può significare "merlo di torre" o "merlo uccello", alla ambiguità nel riconoscimento del "soggetto" di un verbo (es. *Il lupo ha sbranato l'agnello' L'agnello ha sbranato il lupo*), ma noi considereremo il problema su un piano generale. (Per una descrizione dei vari tipi di ambiguità, si veda Parisi e Castelfranchi, in corso di stampa, Cap. 7).

L'intelligenza artificiale tradizionale ci dice che il problema della incertezza sulla strada da prendere in un punto di ambiguità può essere risolto in due modi contrapposti. La soluzione non-deterministica è quella di decidere sempre e subito, scegliendo la strada che si presenta come la più probabilmente corretta. Se questa strada risulterà errata - come è sempre possibile trattandosi di una decisione probabilistica e quindi non sicura al 100. - il sistema dovrà tornare indietro (backtracking), disfare quello che ha fatto e prendere un'altra strada. Questa soluzione non-deterministica è quella più comunenmente accettata in intelligenza artificiale ed è quella adottata nel sistema di comprensione WEDNESDAY (vedi [15]). L'altra soluzione è quella deterministica e comporta un meccanismo di attesa o di "guardare-in-avanti" (lookahead) nel sistema. Se la decisione è incerta e rischia di far sbagliare il sistema, non viene presa nessuna decisione, il sistema aspetta, guarda in avanti sul contesto successivo e decide solo quando ha trovato nel contesto successivo la risposta ai suoi dubbi. In questo modo il sistema non sbaglia mai, quello che costruisce non deve essere mai distrutto e cambiato (tranne casi eccezionali che corrispondono alle frasi in cui gli stessi esseri umani si sbagliano e che quindi vengono trattate non-deterministicamente dal sistema; vedi [8].

Vediamo come questi due approcci affronatno un caso particolare di ambiguità. Si consideri una frase che comincia con le parole *Franco mi...* La parola *mi* costituisce

un punto di ambiguità. A seconda di come la frase viene completata, *mi* può esprimere un caso accusativo *(Franco mi guarda),* un caso dativo *(Franco mi piace)* o un caso benefattivo *(Franco mi legge un libro).* Un sistema deterministico giunto alla parola *mi* deve scegliere praticamente a caso dato che il contesto precedente non dà molta informazione su quale è l'interpretazione giusta di *mi* e dato che le tre interpretazioni di *mi* sono sostanzialmente di uguale frequenza d'uso nel linguaggio. Vi sono quindi due terzi di probabilità che sbaglierà - mentre in una frase che comincia con *Franco mi ...* un essere umano non sembra avere mai la sensazione di sbagliare. Un sistema deterministico si comporta in modo opposto. Visto che al momento in cui incontra *mi* non ha sufficienti ragioni per decidere su una delle tre interpretazioni, rinvia la decisione, aspetta che la frase sia completata e solo allora decide.

Sia l'approccio non deterministico che quello deterministico hanno aspetti insoddisfacenti. L'approccio non deterministico assume che il sistema faccia un grande lavoro che poi viene buttato via, e inoltre non riesce a dar conto del fatto che gli esseri umani hanno la sensazione di sbagliare solo in alcuni casi particolari ma non in casi in cui invece il sistema non deterministico sbaglia facilmente - come nelle frasi che cominciano con *Franco mi ...* che abbiamo visto più sopra. L'approccio deterministico può dar luogo a sistemi molto complicati di attese e rinvii e inoltre sembra smentito da alcuni fatti. (Si veda [9]). Ma a parte questo, il problema più serio è che entrambi i sistemi sono stati sviluppati per trattare l'ambiguità che si presenta all'interno di un analizzatore sintattico (parser), ma non appaiono molto appropriati, né l'uno né l'altro, per trattare l'ambiguità che si incontra nella interazione tra la frase e la conoscenza enciclopedica precedente. In questo secondo caso, le alternative che si presentano non sono ben definite (ad esempio, a cosa può riferirsi un pronome? quale la relazione tra A e B nella espressione A di B, es. *i quadri di Franco?*), i test per verificare se una scelta fatta è giusta o sbagliata (non-determinismo) o per prendere una decisione rinviata (determinismo) non sono essi stessi necessariamente decisivi, e tutto il meccanismo di risoluzione delle ambiguità è globale, continuo e dinamico, piuttosto che locale e discreto.

In effetti, sia l'approccio deterministico che quello non deterministico presuppongono una architettura sequenziale del sistema computazionale, cioè una architettura che potrebbe essere intrinsecamente incapace di supportare sistemi intelligenti come sono quelli di comprensione linguistica. Come di presenta il problema della disambiguazione nell'ambito di un sistema connessionistico e a parallelismo elevato? Torniamo al caso delle frasi che cominciano con *Franco mi ...* Noi assumiamo che la comprensione delle frasi consista nell'attivazione di una parte di una estesa rete di nodi interconnessi che rappresenta tutta la conoscenza del sistema, sia quella sintattica che quella enciclopedica. In particolare possiamo assumere che la rete contenga, tra gli altri nodi, un nodo PARLANTE, e un nodo ciascuno per ACCUSATIVO, DATIVO e BENEFATTIVO. Quando il sistema giunge a processare la parola ambigua *mi,* questa parola attiva il nodo PARLANTE, che è comune alle tre interpretazioni della parola, e attiva anche i tre nodi ACCUSATIVO, DATIVO e BENEFATTIVO, che invece appartengono alle tre interpretazioni alternative di questa parola. Si consideri che nel nostro sistema comprensione significa mettere in relazione, cioè trovare una connessione tra i nodi diversi. Quindi, quando un nodo è attivato, ancora non si ha comprensione; si ha comprensione solo quando è stata trovata una connessione nella rete tra quel nodo e altri nodi anch'essi attivi. (Questo si può riassumere dicendo che una parola isolata non può essere "capita". Può essere capita solo quando cessa di essere una parola isolata, cioè viene messa in relazione con qualcos'altro). Quindi, se si assume che non viene cercata (o trovata) nessuna connessione tra il nodo attivato dalla parola *Franco* e i diversi nodi attivati dalla parola *mi,* la parola *mi* non è ancora veramente "capita" fino a questo punto.

Dopodiché, il sistema procede a processare le parole della frase che vengono dopo *mi.* Poniamo che la frase sia *Franco mi guarda.* La parola *guarda* attiva un modo di GUARDARE che è già connesso nella rete con tre altri nodi, un nodo di CHI GUARDA, un nodo di CHI È GUARDATO. Il nodo di CHI È GUARDATO è a sua volta connesso con il nodo ACCUSATIVO, mentre questo non è vero per gli altri due nodi. A questo punto il sistema cerca il percorso minimo che connette il nodo PORTARE con i nodi attivati in precedenza da *Franco* e da *mi.* La connessione tra *mi* e *portare* passa attraverso il nodo ACCUSATIVO e non attraverso gli altri nodi attivati in alternativa da *mi,* cioè DATIVO e BENEFATTIVO. Pertanto a questo punto il sistema ha disambiguato la parola mi, scegliendo l'interpretazione ACCUSATIVO e non quella DATIVO o BENEFATTIVO. Con queste le altre due frasi, cioè *Franco mi piace* e *Franco mi legge un libro,* sarebbero rispettivamente le interpretazioni DATIVO e BENEFATTIVO a venir scelte con un meccanismo analogo.

Si noti che il meccanismo connessionistico di disambiguazione che abbiamo descritto non è né deterministico né non-deterministico. Non è deterministico perché non comporta nessun "guardare-in-avanti" e nessuna "finestra" predeterminata che consenta di sapere cosa c'è nel contesto successivo quando si deve prendere una decisione in un punto determinato della frase. Non è neppure non-deterministico perché il sistema non fa errori e quindi non fa backtracking. Ad ogni punto della frase il sistema fa quello che può e deve fare, e questo risulta in generale corretto. Questo almeno in tutti i casi in cui gli esseri umani non trovano difficoltà nella comprensione e non hanno la sensazione di aver sbagliato nel processare la frase. Nei casi in cui gli esseri umani hanno difficoltà e sentono di aver fatto un errore, il sistema connessionistico potrà riprodurre questo aspetto della capacità umana di comprensione sulla base di un meccanismo di soglia superata la quale viene presa una decisione che il contesto successivo potrà smentire attivando connessioni alternative. I sistemi connessionistici con le loro caratteristiche di aggiustamento dinamico continuo del pattern di attivazione nella rete, con la loro capacità di incorporare naturalmente elementi non digitali e non discreti - tutte proprietà che abbiamo già notato più sopra - sembrano

particolarmente adatti a riprodurre la complessa microdinamica della comprensione linguistica.

Abbiamo visto come un approccio connessionistico e parallelo alla comprensione del linguaggio consenta di superare l'antinomia tra sistemi deterministici e sistemi non deterministici, entrambi basati sulla visione tradizionale sequenziale dei sistemi computazionali. In questo modo è possibile avere una trattazione unificata di tutta la fenomenologia dell'ambiguità che si incontra nella comprensione linguistica e ottenere la risoluzione dell'ambiguità non tramite speciali meccanismi di cui non ci sarebbe bisogno se nel linguaggio non ci fosse ambiguità e che pongono la difficile scelta tra determinismo e non-determinismo, ma come un semplice sotto-prodotto dei normali processi che avvengono nella comprensione.

Ma, come abbiamo accennato, la questione del determinismo e non-determinismo va al di là dei problemi di comprensione del linguaggio naturale, ed è considerata una questione centrale nella concezione del controllo dei processi in intelligenza artificiale. La possibilità che i sistemi connessionistici offrono di superare tale questione svuotandola di importanza, mostra come il connessionismo sia un modo generale di concepire i sistemi computazionali, che va ai fondamenti della loro costruzione e che rappresenta quindi un paradigma nuovo nella concezione di tali sistemi.

6. MICRO-COGNIZIONE

In realtà il connessionismo spinge a porsi in modo nuovo molti degli stessi problemi epistemologici che sono alla base dell'approccio computazionale. (Si veda la discussione dei problemi generali di scienza cognitiva posti dall'approccio parallelo distribuito nel capitalo 4 di [13]. Su uno di questi problemi vogliamo ora soffermarci per concludere il presente lavoro.

L'approccio computazionale così come finora si è affermato in intelligenza artificiale si è costituito tra l'altro attraverso l'individuazione di un livello di organizzazione e di concettualizzazione che può essere definito nel modo migliore usando il termine di "simbolico". È a questo livello che i sistemi che esibiscono intelligenza vengono studiati dall'intelligenza artificiale. Più sotto di questo livello vi sono altri livelli che non interessano l'intelligenza artificiale e che giungono fino alla struttura e ai processi fisici della macchina che supporta, appunto fisicamente, l'intelligenza. Da questo punto di vista si può capire perchè l'intelligenza artificiale ha scisso i suoi legami con lo studio del cervello, come abbiamo visto in precedenza.

Ora il connessionismo in sostanza non accetta questo punto di vista. Il sottotitolo del fondamentale libro di Rumelhart e McClelland, EXPLORATIONS IN THE MICROSTRUCTURES OF COGNITION, dichiara questo punto esplicitamente. Il tentativo e l'obiettivo del connessionismo è di scendere a un livello più microscopico nello studio della mente, al di sotto del livello simbolico. Più concretamente, le nozioni introdotte e usate in intelligenza artificiale e in psicologia cognitiva, cioè frame, script, schema, prototipo, regole, regole di produzione, ecc. costituiscono un livello macro-teorico che può costituire una prima approssimazione allo studio dell'intelligenza. Tuttavia, una analisi più approfondita dell'intelligenza può richiedere di lavorare a un livello più fine, dove è possibile raggiungere una maggiore omogeneità e semplicità delle teorie (un criterio fondamentale di valutazione delle teorie scientifiche) e l'impiego di meccanismi più fondamentali e elementari (come sono i concetti di nodo, di connessione, di attivazione, ecc.). Tra l'altro, come suggeriscono Rumelhart e McClelland [13] (op. cit. pag. 126), il lavorare a questo livello microcognitivo può spingerci a formulare nozioni macro-cognitive e simboliche diverse da quelle oggi in uso.

Come si vede, una impostazione di questo genere rappresenta un allontanamento significativo da alcune delle assunzioni fondamentali dell'intelligenza artificiale, della psicologia cognitiva e dell'approccio computazionale così come finora si sono sviluppati. Da un lato, non deve sorprendere perchè il connessionismo desideri instaurare rapporti diversi e più stretti con coloro che si occupano del cervello come macchina fisica. Scendendo al di sotto del livello simbolico, è chiaro che prima o dopo si incontrano i livelli a cui sono interessati i neuroscienziati e i fisici. Dall'altro, è evidente il salto epistemologico che il connessionismo può rappresentare applicando per la prima volta in modo sistematico l'approccio computazionale alle microstrutture della cognizione. Su questa strada è possibile sperare di colmare in qualche misura il vuoto ancora esistente tra psicologia e neuroscienze e di pervenire a un livello più profondo di spiegazione delle strutture e dei processi cognitivi studiati dalla psicologia e dall'intelligenza artificiale.

7. CONCLUSIONI

In questo lavoro abbiamo cercato di mostrare come il connessionismo e i sistemi computazionali a parallelismo elevato rappresentino un nuovo paradigma nello studio della mente. Al di là delle sue basi tecnologiche (costruzione di macchine e di software a parallelismo elevato), il connessionismo fornisce strumenti concettuali per impostare in modi nuovi lo studio computazionale dell'intelligenza. Esso consente di riaffrontare con maggiori speranze di successo il problema dei rapporti tra mente e cervello e dello studio computazionale del cervello come organo fisico. Il connessionismo permette anche di costruire sistemi computazionali che posseggano caratteristiche tipiche dell'intelligenza naturale ma che sono state finora riprodotte poco e con difficoltà nei sistemi computazionali dell'intelligenza artificiale tradizionale (sensibilità al contesto, memoria dinamica, prestazioni flessibili), e di impostare in modo nuovo problemi fondamentali dell'intelligenza artificiale e dell'informatica come quello del determinismo-non-determinismo. Infine il connessionismo può significare una svolta sul piano epistemologico concentrando la sua attenzione su un livello dei sistemi computazionali più fondamentale e più elementare rispetto a quello simbolico dell'intelligenza artificiale tradizio-

nale, e consentendo così da un lato di raggiungere un livello di spiegazione più avanzato e dall'altro di stabilire un ponte con lo studio delle basi fisiche della mente.

8. RIFERIMENTI

[1] D.H. Ballard, CORTICAL CONNECTIONS AND PARALLEL PROCESSING: STRUCTURE AND FUNCTION, Behavioral and Brain Sciences, 9, pp. 67-120, 1986

[2] H.L. Dreyfus, WHAT COMPUTERS CAN'T DO, New York: Harper and Row, (edizione riservata), 1979

[3] J.A. Feldman, D.H. Ballard, CONNECTIONIST MODELS AND THEIR PROPERTIES, Cognition Science, 6, pp. 205-254, 1982

[4] D. Hillis, THE CONNECTION MACHINE, Cambridge, Mass., MIT Press, 1985

[5] E.R. John, Y. Tang, A.B. Brill, R. Young, K. Ono, DOUBLE-LABELED METABOLIC MAPS OF MEMORY, Science, 233, pp. 1167-1175, 1986

[6] J.L. Kolodner, C.K. Riesbeck, (eds), EXPERIENCE, MEMORY, AND REASONING, Hillsdale, N.J.: Erlbaum, 1986

[7] T. Kuhn, THE STRUCTURE OF SCIENTIFIC REVOLUTIONS, Chicago, The University of Chicago Press, 1962. (Trad. it. Torino, Einaudi, 1969)

[8] M.P. Marcus, A THEORY OF SYNTACTIC RECOGNITION FOR NATURAL LANGUAGE, Cambridge, Mass., MIT Press, 1980

[9] R.W. Milne, PREDICTING GARDEN PATH SENTENCES, Cognitive Science, 6, 349-374, 1982

[10] D. Parisi, C. Castelfranchi, LA MACCHINA E IL LINGUAGGIO, Torino, Boringhieri, in corso di stampa (a)

[11] D. Parisi, C. Castelfranchi, DISAMBIGUATION IN A LEXICALLY BASED SENTENCE UNDERSTANDING SYSTEM, in S.L. Small, G.W. Cottrell, M.K. Tanenhaus (Eds.), "Lexical ambiguity resolution in the comprehension of human language", in corso di stampa (b)

[12] D. Parisi, S. Nolfi, UN MODELLO CONNESSIONISTICO DELLA COMPRENSIONE DEL LINGUAGGIO, Istituto di Psicologia, C.N.R., 1986

[13] D.E. Rumelhart, J.L. McClelland, PARALLEL DISTRIBUTED PROCESSING: EXPLORATIONS IN THE MICROSTRUCTURES OF COGNITION, Cambridge, Mass., MIT Press, 1986

[14] R.C. Schank, DYNAMIC MEMORY, Cambridge, Cambridge University Press, 1982

[15] O. Stock, C. Castelfranchi, D. Parisi, WEDNESDAY: PARSING FLEXIBLE WORD ORDER LANGUAGES, 1st Meeting of ACL, European Chapter, Pisa, 1983

[16] J. von Neumann, THE COMPUTER AND THE BRAIN, New Haven, Yale University Press, 1958

MODELLI DI RAPPRESENTAZIONE DELLA CONOSCENZA TEMPORALE: RASSEGNA DI LETTERATURA

P. Dell'Orco, R. Giorgesi

FIAR - Milano

RIASSUNTO

Nell'ambito dei Sistemi basati sulla Conoscenza, la rappresentazione del tempo e delle relazioni temporali tra eventi sta assumendo un interesse sempre maggiore.
L'articolo analizza un certo numero di lavori recenti nell'ambito della rappresentazione della conoscenza temporale, classificandoli in una serie di modelli. Per ogni modello sono evidenziate le caratteristiche più significative e vengono identificate le classi di problemi che ciascuno di essi è in grado di rappresentare nel modo più adeguato e completo.

ABSTRACT

While Knowledge-Based Systems are leaving universitary and industrial areas to tackle real-world applications, time representation is more and more becoming a crucial topic.
This paper analyzes recent works in the field of temporal knowledge representation; they are grouped in seven main classes, in accordance with their most distinguish features.
Comments about different class capabilities of facing some significant problems are reported as well.

1. INTRODUZIONE

Nell'ambito dei Sistemi basati sulla Conoscenza, la rappresentazione del tempo e delle relazioni temporali tra eventi, anche di diversa natura, sta assumendo un interessa sempre maggiore.

Questo lavoro è stato parzialmente finanziato dalla Commissione delle Comunità Europee nell'ambito del progetto ESPRIT 932, intitolato "Knowledge-Based Real-Time Supervision in Computer Integrate Manufacturing".

In particolare, i recenti progressi nelle aree che trattano modelli dinamici di realtà complesse, richiedono una rappresentazione della conoscenza temporale e dei fenomeni ad essa correlati molto più potente che nel passato. Nei primi sistemi dedicati alla risoluzione automatica di problemi, non era sentita la neccesità di fornire una modalità di rappresentazione esplicita del tempo: tali sistemi, in genere, operavano su ambienti ridotti - o simulati - non aventi elevate caratteristiche dinamiche (ad esempio, il gioco degli scacchi).

Una certa approssimazione nella rappresentazione del tempo era accettabile in quanto l'interesse dei ricercatori era focalizzato principalmente sulla definizione di architetture adeguate per descrivere formalmente i problemi e sulla individuazione di metodologie adatte alla loro risoluzione.

Con il consolidamento delle tecniche risolutive in Intelligenze Artificiale (IA) e la crescente applicazione dei Sistemi Esperti in ambienti dalle elevate caratteristiche dinamiche, e quindi particolarmente complessi, è nata l'esigenza di una rappresentazione esplicita e accurata delle correlazioni temporali tra eventi. Si cita, per tutti, il campo delle applicazioni biomedicali che storicamente ha visto nascere i primi Sistemi Esperti. In questi sistemi le condizioni dei pazienti sono usualmente descritte in termini di successioni di stati legati da precise relazioni temporali.

Ovviamente sono numerosissimi gli ambiti, di diverse natura e complessità, in cui la rappresentazione esplicita del tempo è particolarmente importante. Tra essi, uno dei più rappresentativi è quello dei Sistemi Produttivi, considerati nello loro globalità.

Le problematiche della "Fabbrica Automatica" (Computer Intergrated Manufacturing) sono attualmente oggetto di ricerche avanzate, che hanno di mira non tanto il solo aspetto dell'automazione, ma anche quello dell'integrazione tra diverse funzionalità: dalla pianificazione della produzione su larga scala, alla programmazione del sistema informativo aziendale, fino alla riprogrammazione in tempi brevi dei robots o delle macchine utensili presenti in officina.

Come è facile intuire, il tempo è un elemento essenziale all'interno di un Sistema Produttivo: flussi decisionali, informativi e di materiali non sono correlati solo "logicamente", bensì devono soddisfare pesanti vincoli temporali, legati alle esigenze produttive (livelli di manodorpera, disponibilità del macchinari, scadenze, ecc.). Oltre a ciò, esistono diversi tipi di informazioni, relative ad eventi particolari - ritardi decisionali e produttivi, guasti di diversa entità - che dovrebbero poter essere rappresentate e gestite nell'ambito di un unico modello.

Nella presente rassegna si definisce "Conoscenza Temporale" (CT) l'insieme delle informazioni relative agli attributi e alle relazioni temporali tra entità - o eventi - in un opportuno sistema di riferimento.

Obiettivo della maggior parte dei lavori esaminati è quello di definire un modello adeguato alla rappresentaezione di tale conoscenza.

Tale modello deve essere in grado di soddisfare una serie di requisiti generali ai fini di una efficiente rappresentazione della conoscenza temporale nelle sue diverse forme. Come sottolinea uno dei più importanti lavori di Allen [1], un modello è adeguato ed efficiente se è in grado di:

1. consentire la rappresentazione di diversi tipi di conoscenza temporale, vale a dire:

   — conoscenza temporale "relativa", in cui entità o eventi sono correlati senza riferimento ad uno specifico intervallo o punto nel tempo.
   Ad esempio:

   Evento__A PRECEDE Evento__B
   o
   Operazione__1 DURANTE Operazione__2

   — conoscenza temporale "assoluta", dove gli eventi vengono correlati ad un particolare periodo di tempo all'interno di un preciso sistema di riferimento.
   Ad esempio:

   Lavorazione L__1 prevista

   dal 1 Dicembre 1986 al 5 Dicembre 1986

   oppure

   Ricovero paziente: 22 Febbraio 1985

   Manifestazione sintomo: ore 17 del 21 Febbraio 1985

   — conoscenza temporale "non omogenea", cioè comprendente riferimenti sia relativi che assoluti:

   Oper-1 PRIMA DI Oper-2 ENTRO il 31 Dicembre 1986

2. operare con informazione "incerta", poiché in molti casi non è possibile definire in modo esatto la correlazione fra due eventi, ma solo un insieme di vincoli relativi alla loro correlazione.
   Ad esempio:

   Lavorazione__1 ora xx

   Lavorazione__2 ora yy

   ma xx <= yy

3. variare gli "ordini di grandezza" temporali ("grain of reasoning"); infatti, l'estrazione di informazioni temporali da una storia (es. biografia) richiede di ragionare in termini di mesi o anni, ma il trattare conoscenza temporale nel settore della progettazione dei computer, implica ragionamenti in termini di millisecondi. Analogamente, in ambito CIM, la pianificazione della produzione su larga scala necessita di avere come "ordine di grandezza" gli anni, mentre i problemi che sorgono a livello di fabbrica (es. pianificazione turni giornalieri) hanno come parametri le ore e i minuti.

Da queste considerazioni emerge che un modello "ideale" deve consentire la rappresentazione della conoscenza temporale a vari "livelli di astrazione", fornendo così uno strumento flessibile per la gestione dei complessi problemi che l'Intelligenza Artificiale - e in particolare l'area relativa alla rappresentazione della conoscenza - si trova ad affrontare attualmente.

2. CLASSIFICAZIONE DEI MODELLI

La letteratura sull'argomento, non molto vasta fino alla fine degli anni Settanta, si è successivamente arricchita con una serie crescente di contributi che testimonia l'interesse destato dal problema. Dall'analisi dei lavori considerati, emergono una serie di approcci, ognuno dei quali soddisfa uno o più requisiti tra quelli specificati nell'introduzione. Ogni approccio è in grado di rappresentare una data classe di problemi.

Nel presentare i lavori considerati, è stata adottata una classificazione già introdotta da Allen [1], basata sulla tipologia del modello di rappresentazione. Non si tratta ovviamente dell'unica modalità di classificazione possibile, ma riteniamo che sia quella che fornisce una visione migliore dello stato dell'arte sull'argomento.

I lavori analizzati sono stati classificati nelle seguenti categorie:

— Spazio degli stati [18], [14], [5], [6];

— Catene before/after [13];

— Date-line [13];

— Intervalli [1], [2], [3], [24], [26];

— Algebra dei punti [26];

— Modelli formali [21];

— Modelli misti [13], [12], [8], [10], [23].

Nel seguito, per ogni modello vengono descritte le principali caratteristiche e le applicazioni più significative.

Allo scopo di evidenziare le caratteristiche di tali modelli, ognuno di essi viene verificato sullo stesso esempio, tratto da uno dei lavori presentati [13]. Tale esempio consiste in una breve biografia:

"Sono nato il 25 Gennaio 1952. Dopo poche settimane fui sottoposto ad una operazione chirurgica; ho iniziato la Scuola Elementare all'età di 5 anni e ho conseguito il Diploma di scuola Media nel Giugno 1965. Due anni dopo il Diploma sono andato in Inghilterra; sono rimasto in Inghilterra per 8 settimane".

L'esempio, pur essendo breve e molto semplice, è ugualmente significativo poiché racchiude tutti i tipi di CT esposti nel paragrafo precedente. Sono infatti individuabili:

— relazioni tra eventi:

  - due anni dopo il diploma sono andato in Inghilterra.

  (esempio di CT relativa)

— relazioni rispetto a un riferimento temporale assoluto:

  - Sono nato il 25 Gennaio 1952

— relazioni non completamente specificate

  - Dopo poche settimane fui sottoposto ad una operazione chirurgica

  - all'età di 5 anni.

2.1 Spazio Degli Stati

Lo spazio degli stati è forse il modello maggiormente utilizzato nei sistemi di consultazione e diagnosi medica. Come sottolineato da Long e Russ nel loro lavoro [18], i medici tendono a ragionare sulle condizioni dei loro pazienti in termini di successione di stati.
In particolare gli autori descrivono un sistema per il controllo in tempo reale delle condizioni di un paziente sottoposto a cure intensive. Un modulo dedicato ha il compito di gestire la conascenza temporale relativa al dominio.

L'evoluzione delle condizioni del paziente è fornita da un flusso di dati, parte dei quali giungono in maniera continua. Un primo passo è quello di estrarre dal flusso "continuo" un insieme discreto di valori, in modo di rappresentare l'evoluzione delle condizioni come una successione di stati.
Una transizione di stato è causata o da una variazione di un dato (ad es. valori della pressione del sangue) o dall'esecuzione di un'azione (ad es. iniezione).

Un particolare trattamento viene riservato a dati che si riferiscono a stati passati. In tal caso si effettua un'analisi retroattiva, partendo dallo stato a cui il nuovo dato si riferisce. In tale analisi le decisioni prese negli stati successivi sono considerate come vincoli (le azioni compiute non possono più essere cambiate anche se esse sono giudicate errate alla luce della nuova analisi).

Le variabili utilizzate dal sistema per trattare tali informazioni sono:

— POINT VARIABLES: variabili che rappresentano fatti veri in specifici stati (es. risultati di analisi di laboratorio o azioni intraprese). La loro rappresentazione interna è costituita da una lista di coppie "informazione-riferimento temporale". La componente temporale di ogni coppia permette di selezionare tutte le informazioni riguardanti un dato intervallo di tempo (stato).

— INTERVAL VARIABLES: variabili che rappresentano informazioni il cui valore si mantiene costante durante un intervallo di tempo. Nell'intervallo possono essere attivati più stati di durata inferiore. Questo tipo di variabile è rappresentato internamente come un insieme di intervalli disgiunti, con un solo valore per ogni intervallo.

— CONTINUOUS VARIABLES: sono utilizzate per rappresentare informazioni estratte da processi continui e, in particolare, per ragionare su dati che riguardano situazioni passate.

Un'altra applicazione interessante in campo medico è il sistema MECS-AI (Medical Consultation System by means Artificial Intelligence) proposto da Koyama [14]. Il sistema si configura come una shell caratterizzata dai seguenti moduli:

— un motore inferenziale;

— un knowledge-base editor per l'acquisizione della conoscenza relativa al dominio di applicazione.

L'aspetto qualificante di MECS-AI è la capacità di trattare dati dipendenti dal tempo. Il fluire del tempo viene rappresentato come una sequenza discreta di eventi. Ogni evento corrisponde all'ingresso di nuovi dati e all'effettuazione di deduzioni su questi dati che si riferiscono all'istante dell'osservazione. Sono definiti alcuni eventi particolari:

— l'evento "target": evento che attiva il processo deduttivo;

— l'evento corrente: l'ultimo evento che è stato creato.

L'informazione temporale relativa a un certo dato è associata ad un attributo della struttura dati utilizzata per memorizzare la conoscenza del dominio. Tale struttura è costituita da una quintupla di attributi (Time-ID), Context, Attribute, Value, Certainty-factor) di cui il primo è quello che racchiude l'informazione temporale, mentre gli altri quattro hanno un significato analogo a quello degli attributi delle strutture presenti in MYCIN.

I valori che può assumere Time-ID sono del tipo: PREVIOUS, IN-PAST, YESTERDAY.

All'inizio il processo di deduzione è attivato dall'evento corrente: evento corrente ed evento target coincidono (figura 1.a). L'attività di deduzione nell'evento target può fare riferimento a dati di eventi precedenti (es. evento II in figura 1.b). Se tali dati non sono disponibili (es. risultati di esami di laboratorio non disponibili al tempo in cui evento II era evento target), il sistema tornerà provvisoriamente all'evento II (evento II = evento target) e dedurrà i dati mancanti (figura 1.c).

Il sistema MECS-AI è stato implementato in INTERLISP e in EPICS-LISP ed è stato utilizzato con successo come sistema di consultazione per malattie della tiroide.

Nell'ambito dei modelli di rappresentazione del tempo classificati nella categoria "Spazio degli stati", va menzionato - con le dovute precauzioni - quello che può essere considerato uno dei più noti formalismi per la rappresentazione del comportamento dinamico di sistemi discreti, vale a dire le Reti di Petri.

Esse sono infatti un potente strumento concettuale per l'analisi e modellizzazione di sistemi caratterizzati da problematiche di sincronizzazione e gestione di processi paralleli. Tale formalismo, introdotto e sviluppato da Petri negli anni 60-62, è uno strumento particolarmente flessibile e generale, che permette una trattazione unitaria di problemi in ambiti anche molto diversi (reti di comunicazione, controllo di processi industriali, gestione di basi di dati distribuite, ecc.).

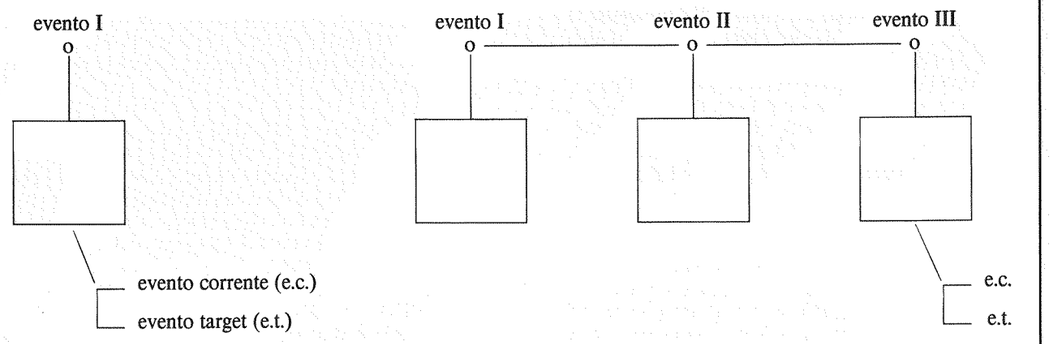


figura 1.a

figura 1.b

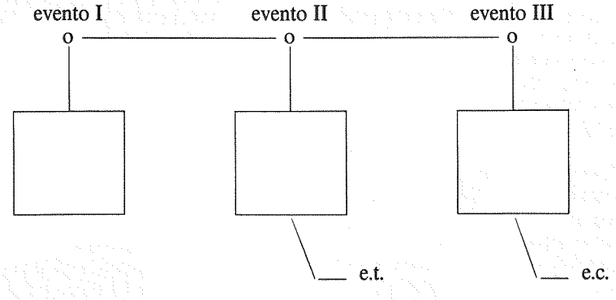


figura 1.c

Fig. 1 Rappresentazione di eventi in MECS-AI (modello "spazio degli stati")

Le Reti di Petri inoltre - con le loro estensioni - costituiscono attualmente il modello formale più avanzato e completo per la descrizione logica delle strutture di controllo del parallelismo. Per una trattazione completa e per la definizione delle entità alla base di tale teoria si rimanda alla letteratura [5].

Le Reti di Petri, o loro sottoclassi, sono state adottate come modello per rappresentare le relazioni tra eventi anche in [11] e [4]. La conoscenza temporale non era esplicitabile perchè le Reti di Petri escludono a priori questa possibilità ma l'associazione ad esse di opportune proprietà, permettono l'individuazione di linee temporali. Inoltre le Reti di Petri sono ottime per modellare situazioni di concorrenza non modellabili facilmente con altri formalismi.

Nell'ambito di questa rassegna si può ricordare molto brevemente che il funzionamento delle reti precisa la dinamica con cui gli eventi modificano lo stato del sistema; è inoltre possibile, data una sequenza di eventi, ricostruire le condizioni che hanno determinato il loro verificarsi. In questo senso, si possono modellizzare processi evolutivi per raffinamenti successivi, pianificare processi con possibilità di restart, gestire processi di diagnostica, ecc.

Quindi, questo significa che, mediante le Reti di Petri, è possibile ordinare eventi nel tempo, decidendo quale evento avviene prima di un altro. Va tuttavia sottolineato che è possibile risolvere conflitti esecutivi (ordinamento sequenziale, cooperazione, ecc.), ma non conflitti strategici; se due eventi possono avvenire contemporaneamente, decidere quale debba avvenire per primo o se debbano essere simultanei, necessità di ulteriori informazioni, e quindi di un'estensione della rete.

Tale formalismo non permette infatti una rappresentazione esplicita delle informazioni temporali associate ad un dato processo rispetto ed un prefissato sistema di riferimento temporale.

Valutazioni

Applicando il modello "Spazio degli stati" alla biografia scelta come esempio di riferimento, si vede che è possibile estrarre le informazioni temporali necessarie a caratterizzare uno stato; la transizione fra due stati è provocata dell'applicazione di un operatore appartenente ad un insieme definito in base a tutte le possibili azioni che si possono verificare durante la vita di una persona.

Rappresentando la biografia con il modello otteniamo lo schema di fig. 2.

Si può osservare che una tale rappresentazione mette in evidenza gli aspetti ordinali fra eventi, ma non fornisce precise informazioni su quando si verifica un evento né sulla sua durata: le correlazioni temporali sono cioè implicite.

Inoltre occorre avere una conoscenza completa dell'ambiente e delle possibili perturbazioni. Il passaggio fra uno stato e l'altro avviene mediante l'applicazione di operatori che alternano lo stato di partenza. È facile intuire che tra dua stati sussiste una relazione di antecedenza: lo stato a cui viene applicato l'operatore precede temporalmente lo stato d'arrivo.

2.2 Catene Before/after

Questo sistema di rappresentazione è stato uno dei primi ad essere proposto e utilizzato [6]. Esso permette di evidenziare le correlazioni temporali relative fra eventi senza però dare loro una collocazione temporale assoluta e senza fornire informazioni sulla durata degli eventi stessi.

Tale modello è una delle rappresentazioni adottate in Time Specialist proposto da Kahn e Gorry [13].
Time Specialist è un modulo dedicato che contiene un insieme di primitive dedicate al trattamento di informazioni aventi riferimenti temporali. Tale modulo può essere incorporato in altri sistemi per trattare la conoscenza temporale presente nel dominio del problema affrontato.

Time Specialist è in grado di trattare frasi contenenti specifiche temporali del tipo:

— Tre settimane fa Giovanni ebbe un raffreddore.

— Giovanni è nato il 6 Giugno 1966.

In queste frasi l'informazione temporale è sempre definita fra almeno due eventi: un evento primario e un evento di riferimento rispetto al quale si colloca temporalmente quello primario. L'evento di riferimento può anche non comparire esplicitamente nella frase. Per esempio nella prima frase, l'evento di riferimento "ora" (momento corrente) è implicito. Nella seconda frase, l'evento di riferimento è il punto zero del calendario. Time Specialist fa una verifica di consistenza delle informazioni temporali estratte. Il sistema accetta un insieme di domande relative a relazioni temporali fra eventi (es. quando un evento si verifica, quali eventi sono accaduti prima di una certa data, ...) ed è inoltre in grado di dedurre il "cammino" fra due eventi.

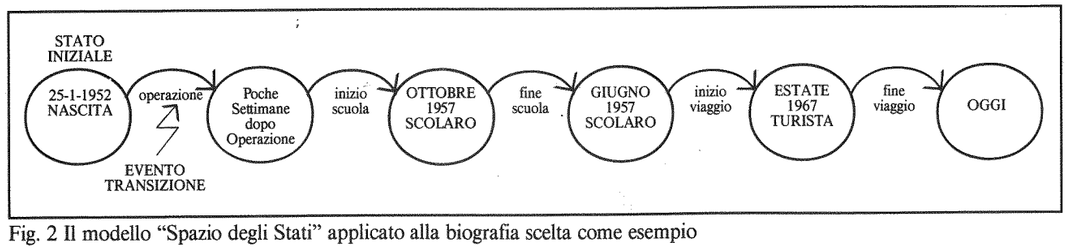


Fig. 2 Il modello "Spazio degli Stati" applicato alla biografia scelta come esempio

Ad esempio, date le specifiche temporali relative agli eventi A, B e C

— A accade cinque giorni prima di B.

— D accade tre giorni dopo B.

— C accade un giorno prima di D.

È possibile incorporare questi eventi in una catena:

(A, B, C, D)

ed includere le informazioni quantitative (es. cinque giorni, ...) in una base di dati.

Così, volendo conoscere l'intervallo di tempo esistente fra gli eventi A e C, per prima cosa occorre trovare il più breve cammino fra A e C per mezzo della catena before/after (cioè (A, B, C)); poi calcolare l'intervallo fra A e C come somma degli intervalli presenti fra le coppie di eventi (A, B) e (B, C) (informazioni ricavabili dal data base).

La rappresentazione della conoscenza temporale relativa alla biografia presentata nell'esempio di riferimento mediante una catena before/after è riportata in fig. 3.

Come si può osservare, tale rappresentazione non fornisce informazioni sulla durata di un evento, né consente di rappresentare correlazioni temporali come, ad esempio, la contemporaneità fra due eventi.

La catena before/after è un modello adatto per rappresentare la conoscenza relativa tra eventi. Esso tuttavia non è in grado di rappresentare alcune particolari relazioni, come ad esempio

"l'evento A e l'evento B sono disgiunti"

Inoltre, quando la dimensione della catena cresce eccessivamente, l'individuazione della correlazione fra due eventi può risultare inefficiente. Infatti il tempo di ricerca di un elemento in una struttura sequenziale è proporzionale alla lunghezza della struttura stessa.
Maggiori considerazioni su questo problema sono presenti in [13].

2.3 Data Line

È una rappresentazione alternativa e complementare a quella precedente, sempre proposta da Kahn e Gorry. Alternativa poiché si presta bene a rappresentare la conoscenza temporale assoluta; complementare poiché, insieme alle catene before/after, permette di catturare tutta la conoscenza temporale, sia relativa che assoluta, presente in un dominio. In Time Specialist la data line è rappresentata da una lista di istanze di eventi, ognuna delle quali include una data e un puntatore alla specifica temporale da cui la data è stata ottenuta o dedotta.

Se un fatto contiene una espressione temporale indeterminata (fuzzy date), ad es. Giugno 1965, il sistema calcola gli estremi dell'intervallo (in questo caso 1/6/1965 e 30/6/1965) e li inserisce nella data line.

La conoscenza temporale estraibile dall'esempio precedente, rappresenta mediante una data line, ha la struttura riportata in fig. 4.

Valutazioni

Una tale rappresentazione, pur non fornendo informazioni dirette sulla durata degli eventi, fornisce:

— la correlazione temporale ordinale fra eventi;

— il momento in cui si verifica un evento;

— l'intervallo esistente fra il verificarsi di due eventi.

Questo modello non è in grado di catturare alcuni tipi di informazioni legate al tempo. Per esempio, la relazione

"l'evento A può accadere prima o dopo l'evento B",

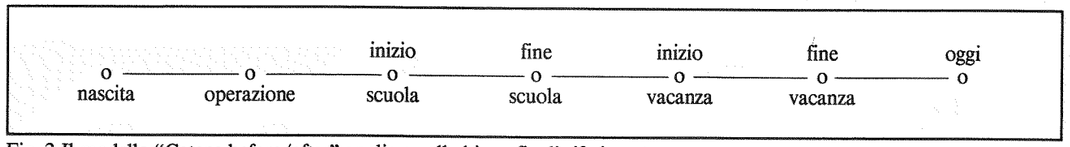


Fig. 3 Il modello "Catene before/after" applicato alla biografia di riferimento

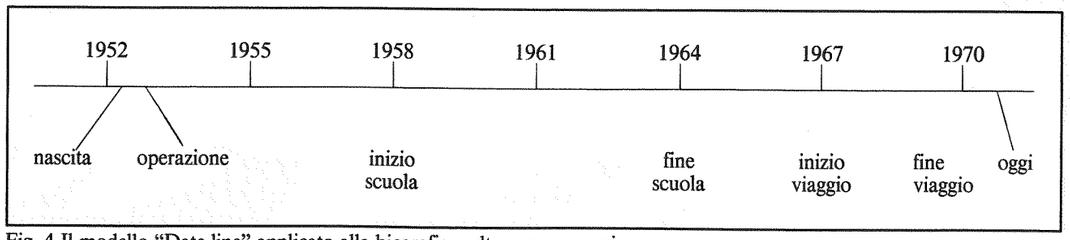


Fig. 4 Il modello "Date line" applicato alla biografia scelta come esempio

INSIEME DELLE POSSIBILI CORRELAZIONI ELEMENTARI

Relazione temporale	Simbolo	Rappresentazione grafica
		BBBBBBBBB
1. A before B	<	AAAAAAA
2. A meets B	m	AAAAAAA
3. A overlaps B	o	AAAAAAA
4. A during B	d	AAAA
5. A finishes B	f	AAAAAA
6. A starts B	s	AAAAAA
7. A equals B	=	AAAAAAAAA
8. A finished-by B	fi	AAAAAAAAAAA
9. A contains B	di	AAAAAAAAAAAAAAAAA
10. A started-by B	si	AAAAAAAAAAAAAAA
11. A overlapped-by B	oi	AAAAAAAAAAA
12. A met-by B	mi	AAAAAAAA
13. A after B	>	AAAAA

Fig. 5 Insieme delle possibili correlazioni elementari definite in [1]

non può essere rappresentata per mezzo di una date line neppure facendo ricorso a una fuzzy date per A e B. (Questo modello è utilizzato da Time Specialist per rispondere a domande su quando un evento accade o sulla durata dell'intervallo esistente fra due eventi).

In conclusione va menzionato il lavoro di Bruce [6], che è stato tra i primi a presentare un modello formale per la rappresentazione della struttura sottostante ai riferimenti temporali nel linguaggio naturale.

2.4 Intervalli

Il modello di rappresentazione del tempo mediante intervalli può essere definito una generalizzazione delle catene before/after: esso infatti risulta appropriato per la risoluzione di problemi in cui la conoscenza temporale è relativa.

L'entità elementare del modello è l'intervallo a cui è associabile un evento. Tra gli eventi è definito un numero limitato di correlazioni elementari, relazioni primitive di Allen [1], che tiene conto anche della durata di ogni evento (fig. 5).

Fra due eventi possono esistere più relazioni primitive. Ad esempio, se la correlazione temporale fra due particolari eventi non è ben definita, essa può essere rappresentata come un insieme di relazioni primitive. Nel momento in cui due eventi si verificano, la relazione temporale che li correla è certamente una di quelle appartenenti all'insieme.

Una esemplificazione di quanto detto può essere tratta dall'ambiente produttivo e riguarda lo scheduling di un processo di fabbricazione di pale per turbine descritto in [23]. Ogni pala viene sottoposta ad un processo di foratura e successivamente ad uno di fresatura. Volendo caratterizzare da un punto di vista temporale le operazioni eseguite su ogni singolo pezzo avremo:

foratura ———(<, m)———> fresatura

Graficamente i processi possono essere rappresentati come illustrato nella figura 6.

La duplice relazione tiene conto delle seguenti possibilità:

— il pezzo viene forato e quindi immediatamente sottoposto a fresatura (macchina fresatrice disponibile);

— il pezzo viene forato e successivamente sottoposto a fresatura (il pezzo fa parte, ad esempio di un lotto di 100 pezzi che viene prima sottoposto a foratura e successivamente a fresatura).

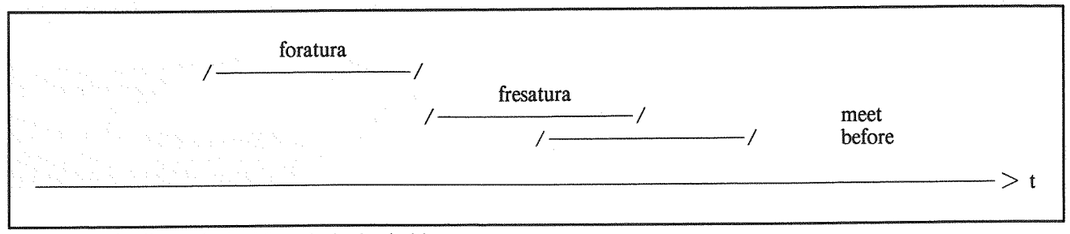


Fig 6 Esempio di correlazioni fra relazioni primitive

Se invece si preferisce rappresentare temporalmente la sequenza di lavorazioni dell'intero lotto di cento pezzi avremo tre possibilità:

foratura ——(<, m, o)——> fresatura

corrispondenti ai tre casi:

— tutti i pezzi del lotto vengono prima forati e in secondo tempo vengono sottoposti a fresatura (la macchina fresatrice non è immediatamente disponibile);

— ogni pezzo del lotto viene forato; completato il processo, il lotto viene inviato immediatamente alla fresatura;

— avviato il processo di foratura, il processo di fresatura inizia non appena sono disponibili dei pezzi del lotto in uscita dalla foratura.

La selezione della relazione temporale effettiva verrà fatta durante la fase di scheduling.

Nel modello di Allen è possibile definire un'algebra degli intervalli basata sulle operazioni di addizione e moltiplicazione.

L'operazione di addizione corrisponde ad una operazione di intersezione fra due insiemi di possibili relazioni temporali. Ad esempio:

ADDIZIONE

Sia

V1 = (before, meets, overlaps)

l'insieme delle relazioni fra due eventi E1 e E2 e

V2 = (overlaps, starts, during)

un ulteriore insieme di relazioni fra E1 e E2.

L'insieme di relazioni che tiene conto di V1 e V2 vale:

V1 + V2 = (overlaps)

Tale operazione viene utilizzata, in genere, quando si definiscono le relazioni fra due eventi partendo da considerazioni indipendenti fra loro: ogni considerazione permette di definire un insieme di relazioni primitive. L'intersezione degli insiemi fornisce l'insieme di relazioni valido fra i due eventi.

Più complessa risulta l'operazione di moltiplicazione. Essa permette, date le relazioni fra le coppie di eventi A, B e B, C di determinare la relazione temporale esistente fra A e C.

MOLTIPLICAZIONE

Sia

V1 = (before, meets, overlaps)

l'insieme di relazioni primitive definite fra due eventi E1 ed E2 e

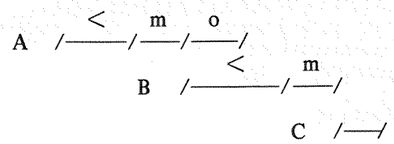
V3 = (before, meets)

l'insieme di relazioni definite fra due eventi E2 ed E3. Il vettore ottenuto dalla moltiplicazione

V1 x V3 = (before)

fornisce l'insieme di relazioni temporali esistenti fra E1 ed E3.

Graficamente:



Per tale operazione è comodo utilizzare la tabella di transitività presente in [1] e riportata in figura 7.

Gli eventi del dominio a cui viene applicato il modello sono rappresentati come nodi di una rete i cui archi sono le correlazioni temporali. Tale organizzazione consente una rappresentazione gerenchica delle correlazioni temporali, e la possibilità di calcolare le relazioni esistenti fra due eventi qualsiasi della rete non direttamente collegati.

L'inserimento di nuovi nodi, o di ulteriori relazioni in un arco già esistente, porta alla definizione di nuove relazioni in cui effetto deve essere propagato su tutta la rete me-

B r2 C A r1 B	<	>	d	di	o	oi	m	mi	s	si	f	fi
"before" <	<	no info	< o m d s	<	<	< o m d s	<	< o m d s	<	<	< o m d s	<
"after" >	no info	>	> oi mi d f	>	> oi mi d f	>	> oi mi d f	>	> oi mi d f	>	>	>
"during" d	<	>	d	no info	< o m d s	> oi mi d f	<	>	d	> oi mi d f	d	< o m d s
"contains" di	< o m di fi	> oi di mi si	o oi dur con =	di	o di fi	oi di si	o di fi	oi di si	di fi o	di	di si oi	di
"overlaps" o	<	> oi di mi si	o d s	< o m di fi	< o m	o oi dur con =	<	oi di si	o	di fi o	d s o	< o m
"over-lapped-by" oi	< o m di fi	>	oi d f	> oi mi di si	o oi dur con =	> oi mi	o di fi	>	oi d f	oi > mi	oi	oi di si
"meets" m	<	> oi mi di si	o d s	<	<	o o s	<	f fi =	m	m	d s o	<
"met-by" mi	< o m di fi	>	oi d f	>	oi d f	>	s si =	>	d f oi	>	mi	mi
"starts" s	<	>	d	< o m di fi	< o m	oi d f	<	mi	s	s si =	d	< m o
"started by" si	< o m di fi	>	oi d f	di	o di fi	oi	o di fi	mi	s si =	si	oi	di
"finishes" f	<	>	d	> oi mi di si	o d s	> oi mi	m	>	d	> oi mi	f	f fi =
"finished-by" fi	<	> oi mi di si	o d s	di	o	oi di si	m	si oi di	o	di	f fi =	fi

Fig. 7 Tabella di transitività tratta da [1]

diante opportuni algoritmi.

La conoscenza temporale dell'esempio scelto come riferimento, mediante questo approccio, verrebbe illustrato come in figura 8.

In figura sono immediatamente identificabili i livelli gerenchici in cui è organizzata la rete. Le relazioni fra l'evento di un livello e quelli a lui correlati di livello inferiore, possono solo essere relazioni la propagazione degli effetti di un successivo inserimento (l'effetto è "visibile" solo nell'intervallo dell'evento di livello superiore, detto evento di riferimento).

Riportiamo ora l'esempio del calcolo di una relazione tra due eventi non direttamente collegati utilizzando la tabella delle transizioni (fig. 9).

Si voglia calcolare la correlazione temporale esistente tra gli eventi:

— fine scuola;

— inizio viaggio.

Prima di tutto occorre calcolare la correlazione tra i due eventi "fine scuola" e "POSTDIPLOMA" indirettamente collegati da "PREDIPLOMA":

T (f, m) = m

Successivamente si calcola la correlazione fra "fine scuo-

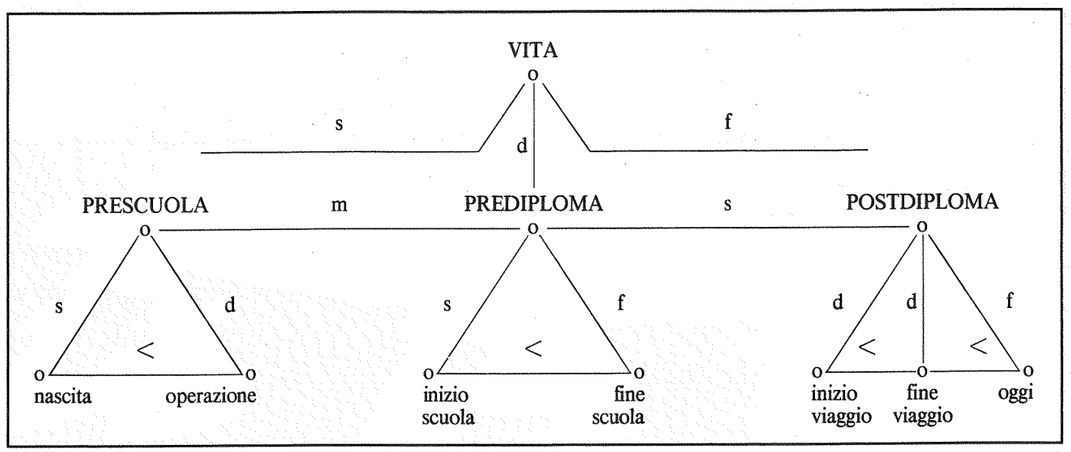


Fig. 8 Rete di eventi rappresentante la biografia d'esempio

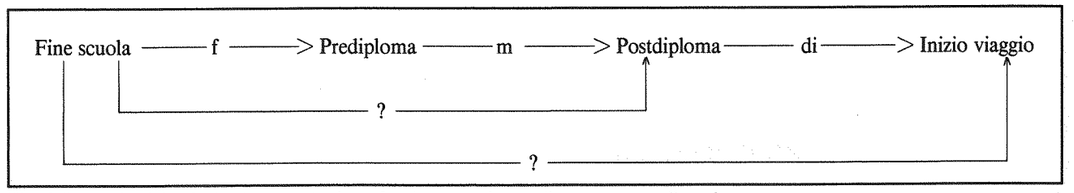


Fig. 9 Esempio del calcolo di una relazione fra due eventi non direttamente collegati

la" e "inizio viaggio" utilizzando la relazione intermedia "POSTDIPLOMA":

T(m,di) = <

che è la relazione cercata.

Va sottolineato che tali algoritmi non sono in grado di rilevare alcuni tipi di inconsistenza presenti nella rete. Essi garantiscono solo la consistenza all'interno di gruppi costituiti da tre nodi della rete, ma non della rete nel suo complesso [1], [24]. Si potrebbe ovviare che diverrebbe di ordine esponenziale, aumentando quindi l'inefficenza degli algoritmi.

Si può comunque dimostrare che il metodo di inserzione e di propogazione di nuovi nodi e/o relazioni non introduce inconsistenze nella rete se le inserzioni sono corrette.

Tale modello di rappresentazione è stato implementato in FRANZ LISP e INTERLISP su VAX 11/780 sotto UNIX. il Modulo è stato utilizzato in alcuni programmi di ricerca, in particolare per la comprensione del linguaggio naturale e in alcuni sistemi per la risoluzione automatica di problemi.

2.5 Time Points

Naturale estensione del modello "date line" è la rappresentazione della CT mediante "time points". In questo modello, l'entità elementare è l'istante di tempo ("time point"): ogni evento viene infatti adeguato definendo i suoi punti d'inizio e di fine.
Ad esempio:

evento: colazione

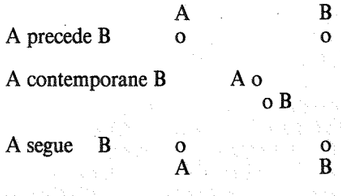
rappresentazione: [iniziocolazione, finecolazione]

o, in forma equivalente

[s__lunch, e__lunch]

con slunch < elunch.

In [26] sono definite le relazioni primitive esistenti fra due punti A e B:



Come si vede, rispetto alla teoria di Allen, le possibili relazioni sono ridotte a tre.

Viene inoltre definita un'algebra dei punti che consente le operazioni di addizione e moltiplicazione. L'addizione è

+	<	<=	>	>=	=	~=	?
<	<	<	0	0	0	<	<
<=	<	<=	0	=	=	<	<=
>	0	0	>	>	0	>	>
>=	0	=	>	>=	=	>	>=
=	0	=	0	=	=	0	=
~=	<	<	>	>	0	~=	~=
?	<	<=	>	>=	=	~=	?

×	<	<=	>	>=	=	~=	?
<	<	<	?	?	<	?	?
<=	<	<=	?	?	<=	?	?
>	?	?	>	>	>	?	?
>=	?	?	>	>=	>=	?	?
=	<	<=	>	>=	=	~=	?
~=	?	?	?	?	~=	?	?
?	?	?	?	?	?	?	?

Key to symbol:

0	is	(), the null vector
<	is	(PRECEDES)
<=	is	(PRECEDES SAME)
>	is	(FOLLOWS)
>=	is	(SAME FOLLOWS)
=	is	(SAME)
~=	is	(PRECEDES FOLLOWS)
?	is	(PRECEDES SAME FOLLOWS)

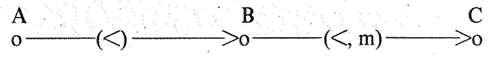
Fig. 10 Tabelle di addizione e moltiplicazione

utilizzata per combinare due differenti insiemi di relazioni fra due eventi: risultato dell'addizione è l'insieme intersezione dei due sistemi.

La moltiplicazione è usata per calcolare le relazioni esistenti fra un evento A e un evento C quando sono note le correlazioni fra A e B e fra B e C (con B evento intermedio fra i due).

In figura 10 sono riportate le tabelle di addizione e moltiplicazione.

Un sistema basato su questo modello consente di rappresentare sia conoscenza temporale relativa che assoluta. Ad esempio l'informazione temporale relativa associata alla rete



può essere rappresentata mediante gli estremi degli intervalli associati agli eventi:

e__A	precede	s__B
e__B	precede, uguale	s__C

con [sA, eA], [sB, eB], [sC, eC] gli estremi degli eventi A, B e C.
Se gli estremi degli intervalli sono associati ad un riferimento assoluto (asse dei tempi) e sono loro assegnati dei valori numerici, l'informazione temporale estraibile è assoluta.

Nel caso di rappresentazione temporale assoluta non sempre è utile, o possibile, rappresentare un intervallo mediante coppia di variabili. Infatti, se la durata dell'intervallo è minore dell'unità di misura adottata sull'asse dei tempi, è sufficiente una sola variabile per rappresentare l'evento.

La rete proposta da Allen per rappresentare le correlazioni tra eventi, viene adattata da Villain in modo da poter operare su di essa in funzione deglie estremi degli eventi stessi. Operazione analoga viene fatta sugli algoritmi che gestiscono la rete.

La conoscenza temporale dell'esempio scelto come riferimento, mediante questo approccio, verrebbe rappresentata come in figura 11.

Gli algoritmi che operano sulla rete e che permettono di:

— calcolare le relazioni fra eventi non direttamente correlati,

— valutare la propagazione dell'effetto dell'inserimento di una ulteriore relazione fra due eventi della rete.

sono computazionalmente meno complessi di quelli che sono stati definiti nel sistema di rappresentazione mediante intervalli. Uno dei fattori che riduce la complessità degli algoritmi è la maggiore semplicità dell'algebra dei punti, basata su tre sole relazioni primitive, rispetto all'algebra degli intervalli, basata su un insieme di relazioni molto più ampio.

Tale modello, comunque, è inadeguato per rappresentare alcune categorie di fenomeni: singoli "time-points" non sono sufficienti ad esprimere la semantica del linguaggio natutrale [26], o a rappresentare correlazioni del tipo "A < B OR A > B" (mutua esclusione).

Un'applicazione di questo modello descrive un sistema per la gestione di informazione temporale e spaziale rappresentabile mediante vincoli lineari. Le informazioni relative agli intervalli sono trasformate in relazioni (diseguaglianze) relative ai loro estremi.

Esempi di correlazioni rappresentabili mediante vincoli lineari sono:

— Giovanni è nato nel 1955

— Carlo è in vacanza da sei giorni

Correlazioni non rappresentabili mediante vincoli lineari sono:

— Giovanni ha incontrato Maria poco fa

— Carlo si laureerà fra due o tre anni.

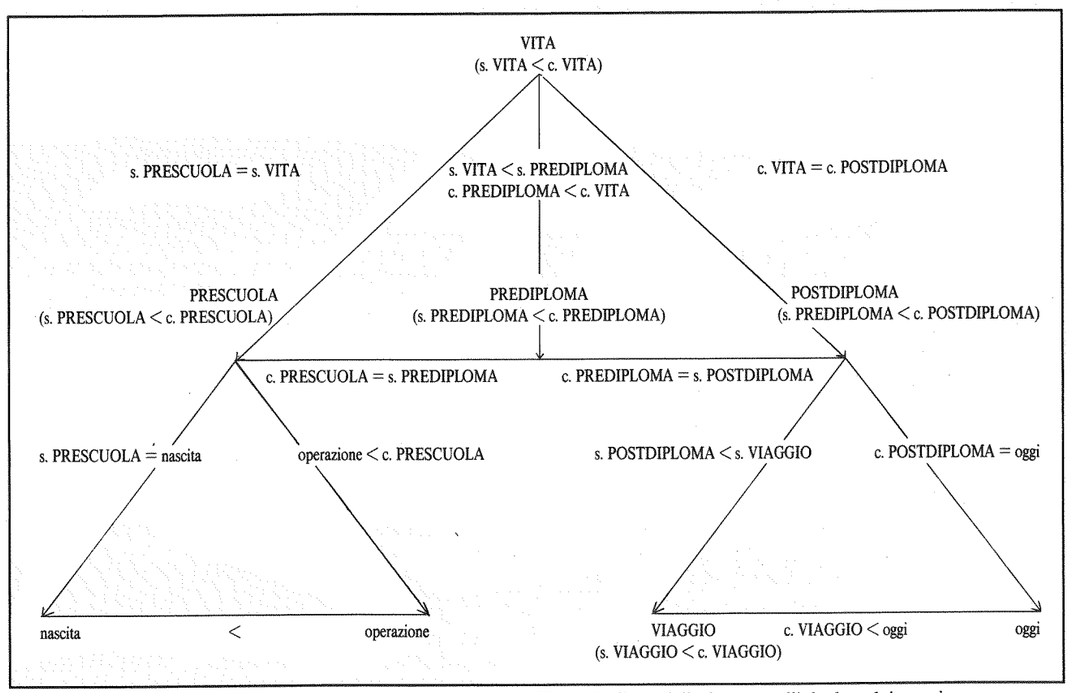


Fig. 11 Rappresentazione della biografia d'esempio ottenuta utilizzando il modello basato sull'algebra dei punti.

Gli eventi direttamente correlati sono raggruppati in un'unica struttura, detta "cluster", con una propria unità di misura temporale. All'interno di ogni struttura, il problema di estrarre informazione temporale relativa agli eventi è ridotto ad un problema di pragrammazione lineare, risolubile col metodo del simplesso (1). Le strutture sono collegate fra loro mediante trasformazioni lineari.

Gli algoritmi che operano sui cluster risultano particolarmente efficienti poichè, operando su un sottoinsieme dell'intera rete, utilizzano sistemi linae di dimensioni accettabili.

Si può dimostrare che il sistema è formalmente adeguato, cioè è consistente e coerente, e contiene meccanismi sufficienti per rappresentare tutte le specifiche temporali ed effettuare tutte le deduzioni possibili.

Tale programma trova applicazione in sistemi che operano in domini in cui la conoscenza di caratteristiche geometriche e temporali è essenziale e interconnessa: una rappresentazione uniforme di tali caratteristiche risulta essere vantaggiosa. Tali caratteristiche sono riscontrabili in campi quali quello della visione e della robotica.

(1) L'algoritmo del simplesso è un metodo proposto da Tucker per risolvere problemi di programmazione lineare. Un problema, tipicamente di ottimizzazione, viene reso equivalente al problema di minimizzare una funzione, tenendo conto di una serie di vincoli lineari che devono essere soddisfatti dalle variabili della funzione stessa. L'algoritmo è facilmente automatizzabile.

2.6 Modelli Formali

Nell'ambito di questa rassegna, si ritiene significativa una breve esposizione del modello proposto da D. McDermott [21]. Esso infatti presenta un modello abbastanza generale, primo nella sua sistematicità, in grado di rappresentare l'informazione temporale in domini di interesse per applicazioni di A1. Una esposizione completa si rimanda alla bibliografia. Partendo dal "situation calculus" [20], viene sviluppato un formalismo basato sulla logica dei predicati del primo ordine che è in grado di rappresentare la conoscenza temporale.

Secondo l'autore alcune caratteristiche tipiche di un dominio reale, che un modello deve essere in grado di rappresentare:

— INCERTEZZA SUGLI STATI FUTURI: a partire da un certo istante possono accadere diversi fatti; ciascuno di essi, a sua volta, può evolvere in molti modi: ovviamente uno solo è quello che si verifica.

— CONTINUITÀ DEL TEMPO: la maggior parte dei processi nel mondo reale evolve in maniera continua.

Le entità temporali del modello utilizzate per fissare queste caratteristiche sono:

— STATO: rappresentazione del mondo reale ad un certo istante. Attraverso l'impiego di operatori logici, è possibile definire relazioni di ordinamento fra gli stati. Ad ogni stato è associabile un valore reale, la data, che indica quando esso si verifica rispetto ad un riferimento assoluto.

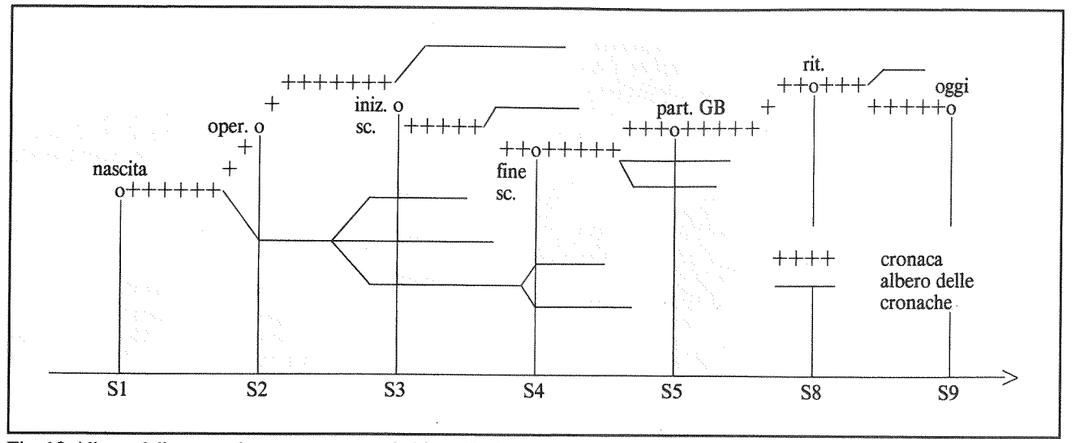


Fig. 12 Albero delle cronache rappresentante la biografia d'esempio

— INTERVALLO: insieme ordinato e finito di stati.

— CRONACA: insieme ordinato di stati che si estende infinitamente nel tempo. Ogni stato dell'insieme è quindi referenziabile in termini sia di CT relativa (mediante gli operatori logici) che assoluta. Poiché il futuro è indeterminato, le cronache possono solo ramificarsi nel futuro (date certe condizioni, esiste un insieme, anche infinito, di possibili eventi che si possono verificare).

Gli eventi che vengono considerati permettono di definire i FATTI (insieme di stati in cui il fatto risulta essere vero) e gli EVENTI (insieme ordinato di intervalli in cui l'evento si verifica).

Utilizzando questo modello, la biografia scelta come esempio verrà rappresentata mediante una cronaca "ramo" di un albero di cronache. In figura 12 è rappresentata la situazione all'istante

$$t = (d\ s9)$$

dove d è la funzione che restituisce la data associata all'evento s9.

Le ramificazioni presenti agli istanti precedenti t indicano le alternative che si sono presentate: di esse solo una è entrata a far parte della cronaca.

Dalla cronaca è estraibile una serie di informazioni temporali relative agli eventi in essa contenuti:

— Informazione temporale assoluta:

(d s1) = t1 istante in cui accade s1. Operazione valida per ogni evento

(< (d s1) (d s2) (d s3) (d s4) (d s5) (d s8) (d s9) ordinamento degli eventi rispetto ad un riferimento assoluto.

— Informazione temporale relativa facilmente ottenibile dalle informazioni già dedotte:

(< s1 s2 s3 s4 s5 s8 s9)

Come già sottolineato, il modello non è generale, cioè valido per qualunque dominio, ma è orientato alla rappresentazione di alcune categorie di problemi. In particolare il modello permette di:

— Individuare relazioni di causalità;

— Ragionare su processi che evolvono nel tempo in modo continuo;

— Eseguire la pianificazione di azioni.

Da un punto di vista implementativo, occorre avere delle strutture dati in grado di gestire cronache multiple: una storia può infatti evolvere in un numero infinito di modi. Ovviamente non è necessario memorizzare tutte le possibili alternative, ma solo quelle che risultano significative rispetto agli obiettivi.

A tale scopo è definito il "chronset", struttura dati in grado di descrivere sia un insieme parzialmente ordinato di cronache sia i meccanismi per poter accedere agli eventi e ai fatti che si verificano all'interno delle cronache.

Per un efficiente rappresentazione dei chronset, l'utilizzo dei "frames" sembra particolarmente indicato.

2.7 Metodi Misti

Ogni modello illustrato finora possiede caratteristiche che

permettono di gestire particolari forme di conoscenza temporale e di operare ad un certo livello di astrazione. Nessuno di essi tuttavia si può definire "completo", in grado cioè di rappresentare tutti i tipi di correlazioni temporali.

Per questo motivo sono stati proposti alcuni modelli, detti appunto "misti", il cui obiettivo è:

1. rappresentare tutta la conoscenza temporale estraibile da un dato dominio;

2. fornire un modello al livello di astrazione in cui si opera.

Tale approccio utilizza concetti già presenti nei modelli esposti in precedenza (punti, intervalli, catene ...), integrandoli mediante definizione delle funzioni che collegano fra loro tali concetti.

In tale ambito è inquadrabile il lavoro [12]. Nel modello proposta (t-model), sono definiti quattro oggetti:

— punti

— intervalli

— catene

— quantità

e i quattro gruppi di relazioni che permettono di operare su di essi.

Sui PUNTI sono definite relazioni logiche di <, <=, =, != (not equal), e la funzione di distanza (R{x1, x2}).

Gli INTERVALLI sono definiti come una coppia ordinata di punti:

$$t = (x1, x2)$$

Le relazioni fra intervalli sono esprimibili sia in termini di punti (estremi degli intervalli stessi) sia utilizzando un insieme di relazioni primitive analogo a quello di Allen.

Ad esempio:

during (t1, t2) <===> (x2 <= x1) AND (yl <= y2)

con t1 = (x1, y1) e t2 = (x2, y2).

Le CATENE sono sequenze di intervalli collegati fra loro da relazioni di antecedente/successivo. Ad ogni catena è associata una serie di attributi che forniscono informazioni sulle sue caratteristiche: numero di elementi, condizioni di appartenenza, proprietà metriche. Un insieme di operatori permette di estrarre informazioni sulla struttura delle catene.

Ad esempio, dati due elementi della catena S, l'operatore:

R-number (S, t1, t2, k)

fornisce la distanza relativa k fra i due elementi t1 e t2.

La QUANTITÀ è il valore della funzione distanza. Su di essa sono definiti operatori che consentono di definire l'unità di misura da utilizzare (es. anni, ore, minuti, ...) e di effettuare trasformazioni fra unità di misura (es. 3 anni e 5 mesi =< 41 mesi). Sono anche definiti operatori che correlano tra loro catene.

Il "t-model" consente di rappresentare la conoscenza temporale presente in diversi campi applicativi (durata di eventi, frequenza, ecc.).

Il modello è stato implementato e utilizzato in programmi per il riconoscimento del linguaggio naturale e di medellizzazione geometrica (spaziale).

Anche "Time Specialist" [13] nel suo complesso può essere inquadrato in questo ambito. Esso infatti permette differenti livelli di rappresentazione (date line, catene before/after, eventi di riferimento) correlati fra loro.

Da ultimo, ci sembra opportuno illustrare - se pur brevemente - un modello sviluppato nell'ambito del progetto ISIS (Intelligent Scheduling and Information System) alla Carnegie Mellon University. (Per una trattazione completa si rimanda alla bibliografia [23], [8], [9], [10].

Il lavoro esamina il ruolo della conoscenza temporale nella pianificazione della attività produttiva con un particolare riguardo:

— alla rappresentazione di processi produttivi (evidenziando le connessioni temporali tra i componenti di tali processi);

— alla rappresentazione e alla gestione di vincoli tempovarianti;

— alla allocazione delle risorse;

— alla gestione di perturbazioni.

Considerando che una singola "prospettiva temporale" è inadeguata per un sistema che deve affrontare questi problemi, viene proposta dagli autori una struttura che permette di definire rappresentazioni temporali multiple fra loro connesse.

Il modello viene descritto utilizzando SRL, Schema Representation Language, (con cui è stato implementato) un linguaggio "frame based" adatto alla rappresentazione di questo genere di conoscenze [9].

Gli elementi fondamentali utilizzati per rappresentare le caratteristiche temporali delle varie entità presenti nel dominio applicativo sono [8]:

— la TIME LINE, che specifica un sistema di rifermento temporale assoluto, ed è definita mediante il seguente schema SRL:


```
{{  time-line
      POINT-FORM:
      START-POINT:
      END-POINT:
      GRANULARITY:
      ADD:
      DIFF:
      BEFOREP:
      AFTERP:
      ...
 }}
```

Gli attributi associati alla struttura forniscono informazioni sulla "granularità" (cioè sull'unità di misura) della time-line e sull'insieme di operazioni che è possibile effettuare su di essa.

— l'INTERVALLO, che definisce il periodo di tempo in cui si verifica un certo evento.

```
{{  interval
        START-LINE:
        END-LINE:
        DURATION:
        DATED-BY:
 }}
```

Un particolare sistema di riferimento assoluto si ottine definendo una istanza di time-line. Ad esempio:

```
{{  calendar
        {ISTANCE time-line
        ...
        START-POINT: (0 0 0)
        END-POINT: (99999 0 0)
        GRANULARITY: (0 0 1)
        ...
        } }}
```

è l'istanza definita da una tripla ordinata di valori (settimana, giorno, ora) e granularità 1 ora.

È possibile associare una classe di intervalli ad una specifica istanza di time-line mediante lo schema:

```
{{  calendar-interval
      {IS-A interval
       DATED-BY: calendar} }}
```

che definisce una classe di intervalli chiamata calendar-interval. Ogni elemento della classe, essendo un intervallo (IS-A interval), erediterà tutti gli attributi del tipo interval. Tale classe è associata mediante la relazione DATED-BY alla time-line calendar. Gli attributi della relazione DATED-BY forniscono una serie di informazioni supplementari sul tipo di legame.

Poiché l'attività di scheduling richiede di trattare CT relativa - capace di rappresentare relazioni tra attività ed eventi - nel modello sono incorporate le relazioni temporali di Allen.
Nel seguito viene riportato un esempio di come la relazione di Allen "during" è tradotta dallo schema SRL:


```
{{  during
    {  IS-A relation
          DOMAIN: (TYPE INSTANCE

          RANGE: (EQ domain)
          INVERSE: contains } }}
```

L'insieme di oggetti e relazioni definite nel modello consente di trattare tutta l'informazione temporale presente nel dominio applicativo.

Come esempio di applicazione ad un caso concreto [23], viene brevemente presentato il problema della pianificazione dei turni di lavoro ("shift") previsti per una macchina utilizzata in un processo di produzione di pale per turbine. La pianificazione copre un arco di 80 settimane.

Il turno viene rappresentato mediante lo schema "shift", una cui istanza è:

```
{{  mach1-5day-1st-shift
    { INSTANCE shift
      START-TIME: (7 30)
      END-TIME: (15 30)
      FOREMAN: Jones
      WORK-WEEK: 5day-workwk } }}
```

I vincoli che caratterizzano temporalmente i turni lavorativi su mach1 sono:

```
{{  mach1-shift-constraint-root
    { INSTANCE shift-constraint-root
      SPECIALIZATION-OF: milling-area-shift-
                                  constraint-root
      ALTERNATIVES: mach1-shifts
      CURRENT-ALTERATIONS: mach1-shift-
                                  constraint1
                           mach1-shift-
                                  constraint1   }}

{{  mach1-shift-constraint1
    { INSTANCE shift-constraint
      TEMPORAL-SCOPE: {{ INSTANCE calendar-
                                  interval
                         START-TIME:
                                  (0 0 0)
                         END-TIME:
                                  (20 0 0) }}
      SHIFT: mach1-5day-1st-shift
      UTILITY: 2
      ALTERATION-OF: mach1-shift-constraint-
                                  root } }}

{{  mach1-shift-constraint2
    { INSTANCE shift-constraint
      TEMPORAL-SCOPE: {{ INSTANCE calendar-
                                  interval
                         START-TIME:
                                  (50 0 0)
                         END-TIME:
                                  (80 0 0) }}
      SHIFT: mach1-5day-1st-shift
      UTILITY: 1.4
      ALTERATION-OF: mach1-shift-constraint-
                                  root } }}
```

Utilizzando una notazione grafica, la rappresentazione e del tipo:

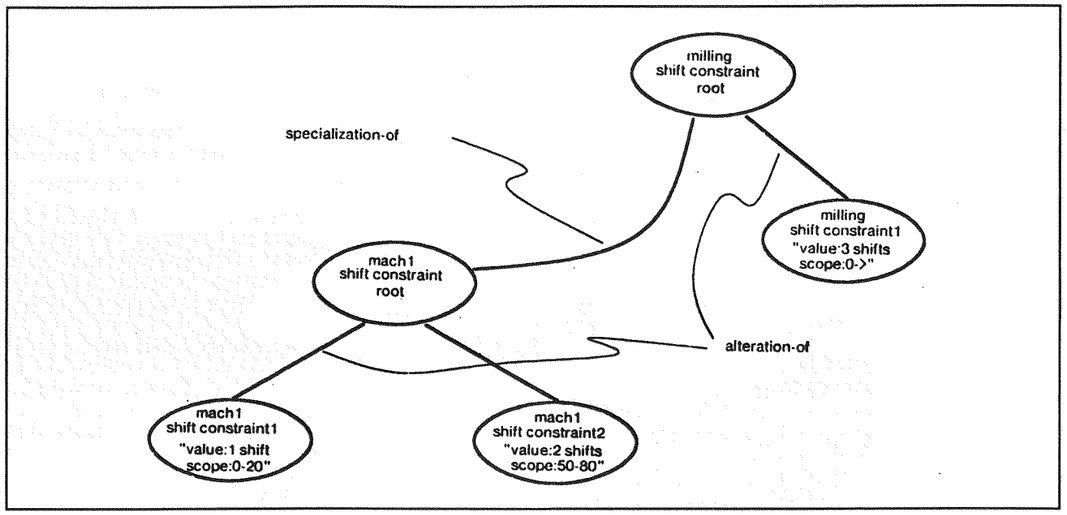


Fig. 13 Rappresentazione grafica della configurazione iniziale dei vincoli sui turni per la macchina mach1 tratta da [23]

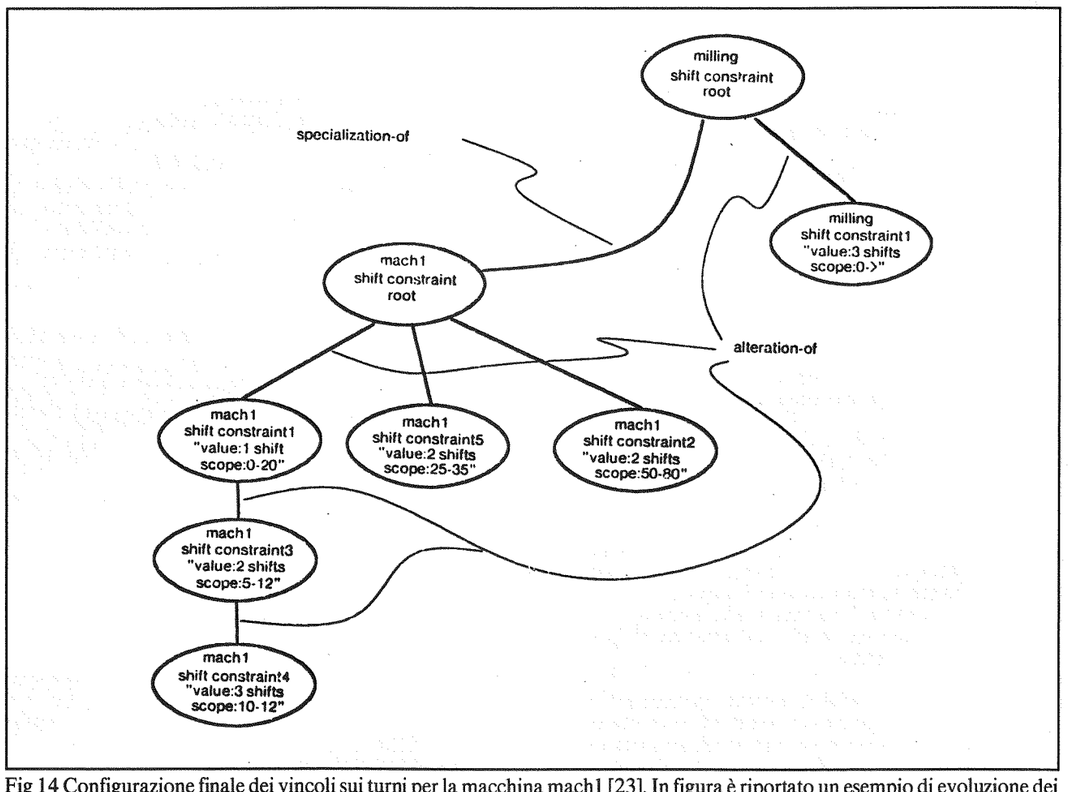


Fig 14 Configurazione finale dei vincoli sui turni per la macchina mach1 [23]. In figura è riportato un esempio di evoluzione dei vincoli su mach1.

Nell'arco delle 80 settimane vi sono vincoli significativi solo nei due intervalli (0, 20), (50, 80) in cui sono richiesti specifici turni di lavoro.

Si supponga che, ad un certo istante, sia presa una decisione che come conseguenza, provoca una variazione dell'attività stabilita su mach1.
L'effetto sarà quello di definire un ulteriore vincolo, che verrà posto nello slot CURRENT-ALTERATION dello schema mach1-shift-constraint-root. Questo vincolo sarà attivo nell'intervallo di tempo che corrisponde al suo periodo di applicabilità, mentre il vincolo originale sarà ancora applicabile nei punti esterni a tale intervallo. Ovviamente un'alterazione può, a sua volta, essere modificata, creando così una struttura ad albero.

Al fine di localizzare il vincolo attivo all'istante t, verrà identificato il vincolo più basso internamente all'albero il cui intervallo di applicabilità contiene t.

3. CONCLUSIONI

La rassegna qui presentata, pur non avendo la pretesa di essere esaustiva, dimostra l'esistenza di un forte interesse per il problema della rappresentazione della conoscenza temporale. È però altrettanto evidente che una soddisfacente soluzione ad esso non è ancora stata trovata.

I modelli proposti sono diversi, ma ognuno, pur essendo adatto a rappresentare in modo adeguato una certa classe di problemi, non è in grado di esprimere tutta la conoscenza presente in molti domini applicativi reali.
Questo comporta la necessità di adottare modelli complessi che incorporino le caratteristiche di più modelli elementari: tale tendenza, emerge effettivamente dall'analisi dei lavori più recenti presenti in rassegna. L'articolo di Smith [23] è, da questo punto di vista, molto significativo.

Gli autori sono attualmente impegnati nella definizione della caratteristiche di un modello "ideale" che soddisfi i requisiti tipici dell'ambiente CTM. Questa definizione è svolta nell'ambito del Progetto ESPRIT 932, in cui la FIAR (Dipartimento di Intelligenza Artificiale) è coinvolta.

Il modello sviluppato nel progetto sarà validato mediante l'implementazione di un prototipo per simulazione di operazioni di scheduling e pianificazione.

4. RINGRAZIAMENTI

Si ringrazia l'Ing. M. Borghesi per la preziosa collaborazione.

5. BIBLIOGRAFIA

[1] J.F. Allen, MAINTAINING KNOWLEDGE ABOUT TEMPORAL INTERVALS, Communications of ACM, vol. 26, n. 11, 832-843, 1983

[2] J.F. Allen, TOWARDS A GENERAL THEORY OF ACTION AND TIME, Artificial Intelligence 23, 2, 1984

[3] J.F. Allen, P.J. Hayes, A COMMON-SENSE THEORY OF TIME, Proceedings of 9th Int. Joint Conference on Artificial Intelligence, 528-531, Los Angeles, 1985

[4] S. Bandini, L. Spampinato, SISTEMI ESPERTI E RAPPRESENTAZIONE ETEROGENEA DELLA CONOSCENZA IN MEDICINA: L'ESEMPIO DELL'URGENZA TOSSICOLOGICA, RAYS, International Journal of Radiological Sciences, Suppl. vol. II, Roma, 1986

[5] G.W. Brams, LE RETI DI PETRI: TEORIA E PRATICA, vol. 1, Masson, 1985

[6] B.C. Bruce, A MODEL FOR TEMPORAL REFERENCES AND ITS APPLICATION IN A QUESTION ANSWERING PROGRAM, Artificial Intelligence 3, 1-20, 1972

[7] Chun Hun Wai, A REPRESENTATION FOR TEMPORAL SEQUENCE AND DURATION IN MASSIVELY PARALLEL NETWORKS: EXPLOITING LINK INTERACTIONS, Proceedings of 5th National Conference on Artificial Intelligence, 372-376, AAAI-86, Philadelphia, 1986

[8] M.S. Fox, B.P. Allen, S.F. Smith, G.A. Strohm, IBIS: A CONSTRAINT DIRECTED REASONING APPROACH TO JOB SHOP SCHEDULING, SYSTEM SUMMARY, Technical Report, The Robotics Institute, Carnegie Mellon University, 1983

[9] M.S. Fox, J.M. Wright, D. Adam, EXPERIENCES WITH SRL: AN ANALYSIS OF A FRAME-BASED KNOWLEDGE REPRESENTATION, Technical Report, The Robotics Institute, Carnegie Mellon University, 1985

[10] S.M. Fox, S.F. Smith, Peng Si Ow, CONSTRUCTING AND MAINTAINING DETAILED PRODUCTION PLANS: INVESTIGATIONS INTO DEVELOPMENT OF KNOWLEDGE-BASED FACTORY SCHEDULING SYSTEMS, AI Magazine, Fall 1986

[11] M. Gallanti, G. Guida, L. Spampinato, A. Stefanini, REPRESENTING PROCEDURAL KNOWLEDGE IN EXPERT SYSTEMS: AN APPLICATION TO PROCESS CONTROL, Proceedings of IJCAI-85, Los Angeles, 1985

[12] E. Yu Kandrashina, REPRESENTATION OF TEMPORAL KNOWLEDGE, Proceedings of IJCAI 83, 346-348, 1983

[13] K. Kahn, G.A. Gorry, MECHANIZING TEMPORAL KNOWLEDGE, Artificial Intelligence, 9, 87-108, 1977

[14] K. Koyama, S. Kaihara, T. Minamikawa, T. Kurokawa, TIME-ORIENTED FEATURES FOR MEDICAL CONSULTATION SYSTEMS, 910-912, Proc. 8th Int. Joint Conference on Artificial Intelligence, Karlsruhe, 1983

[15] P. Ladkin, PRIMITIVES AND UNITS FOR TIME SPECIFICATION, in Proceedings of the 5th National Conference on Artificial Intelligence, 354-359, AAAI-86, Philadelphia, 1986

[16] P. Ladkin, TIME REPRESENTATION: A TAXONOMY OF INTERVAL RELATIONS, in Proceedings of the 5th National Conference on Artificial Intelligence, 360-366, AAAI-86, Philadelphia, 1986

[17] B. Leban, D.D. McDonald, D.R. Forster, A REPRESENTATION FOR COLLECTIONS OF TEMPORAL INTERVALS, in Proceedings of the 5th National Conference on Artificial Intelligence, 367-371, AAAI-87, Philadelphia, 1986

[18] W.J. Long, T.A. Russ, A CONTROL STRUCTURES FOR TIME DEPENDENT REASONING, 230-232, Proc. 8th Int. Joint Conference on Artificial Intelligence, Karlsruhe, 1983

[19] J. Malik, T.O. Bindford, REASONING IN TIME AND SPACE, 343-345, Proc. 8yh Int. Joint Conference on Artificial Intelligence, Karlsruhe, 1983

[20] J. McCarthy, P. Hayes, SOME PHILOSOPHYCAL PROBLEMS FROM THE STANDPOINT OF ARTIFICIAL INTELLIGENCE, Machine Intelligence, 4, Edinburgh University Press, Edinburgh, 1983

[21] D. McDermott, A TEMPORAL LOGIC FOR REASONING ABOUT PROCESSES AND PLANS, Cognitive Science 6, 101-155, 1982

[22] J.F. Rit, PROPAGATING TEMPORAL CONSTRAINTS FOR SCHEDULING, in Proceedings of the 5th National Conference on Artificial Intelligence, 383-388, AAAI-86, Philadelphia, 1986

[23] S.F. Smith, EXPLOITING TEMPORAL KNOWLEDGE TO ORGANIZE CONSTRAINTS, Technical Report, The Robotics Institute, Carnegie Mellon University, 1983

[24] E.P.K. Tsang, PLAN GENERATION IN A TEMPORAL FRAME, vol. 1, 479-493, 7th European Conference on Artificial Intelligence, Brighton, 1986

[25] J.K. Tsotsos, TEMPORAL EVENT RECOGNITION: AN APPLICATION TO LEFT VENTRICULAR PERFORMANCE, 900-907, Proc. 7th Int. Joint Conference on Artificial Intelligence, Vancouver, 1981

[26] M. Vilain, H. Kautz, CONSTRAINT PROPAGATION ALGORITHMS FOR TEMPORAL REASONING, in Proceedings of the 5th National Conference on Artificial Intelligence, 377-382, AAAI-86, Philadelphia, 1986

INTEGRATING GEOMETRIC AND FUNCTIONAL KNOWLEDGE IN COMPUTER VISION

Adriana Batistoni, Mauro di Manzo, Franca Ricci, Emanuele Trucco

Dipartimento di Informatica, Sistemistica e Telematica

Genova

RIASSUNTO

Il riconoscimento di oggetti nella visione artificiale è sostanzialmente basato sul confronto dei dati estratti dalle immagini con classi di prototipi. Un problema centrale è il tipo di conoscenza che costutuisce i prototipi stessi, e che viene impiegata per guidare il processo di riconoscimento. Una relazione esclusivamente geometrica si rivela spesso inadatta a fronte della varietà dei possibili oggetti di una stessa classe. L'uso di conoscenza funzionale in un sistema di visione può rendere più flessibile la descrizione degli oggetti. Essa lega inoltre la struttura degli oggetti alla funzione cui sono dedicati. In questo lavoro si presenta un modello di descrittori semantici basato su primitive funzionali ed alcuni risultati ottenuti da una prima implementazione.

ABSTRACT

Object recognition in computer vision is basically a matching process, where data extracted from images and variously processed are matched against stored models. A central point is what kind of knowledge should be embedded in object descriptors, which will be used to drive the recognition process. A fully geometric description can hardly cope with the many different shapes that an object can assume. Using functional information can improve the flexibility of the system. Moreover, it links structural description of objects to their functionality. In this paper we present a semantic model based on functional primitives and some results of its first implementation.

1. INTRODUCTION: THE SHAPE VS. FUNCTION DEBATE

The problem of object recognition in vision can be approached from different points of view. A well known approach in computer vision and graphics suggests to classify objects according to their geometric properties (see for instance [1,17]. Object recognition is basically interpreted as some sort of geometric pattern matching. Complex objects are usually represented according to some decomposition criteria (e.g. in ACRONIM [4]; see also [16]). Facing high-level vision tasks, however, this geometry-based approach appears to suffer from two main drawbacks. On the one hand, the object representation techniques currently employed either specialize in a particular class of shapes or ignore geometric details in favor of more abstract information (see Bels and Jain [1] for a recent detailed discussion). On the other hand the description of all the possible variations of an object shape is quite an expensive computational task. If the model cannot grow indiscriminately, a shape-based system is likely to consider only a limited number of different instances for each given object.

To avoid the explosion of the data base, one could think to describe an object through its function. The starting point is the observation that the shape of man-made objects is closely related to the task they are designed to accomplish. They could therefore be represented by a set of functional constraints better than by simple structure descriptions. According to this hypothesis, objects would not be defined necessarily by a prototypical shape or a composition of substructures but through a set of functional constraints. A chair would be consequently regarded as "something on which you can sit on without falling back", a hammer as "something you can grasp to hit things" and so forth.

Seminal work in this field has done by Freeman and Newell [11] and, more recently, the approach has been taken by Ingrand and Latombe [12], Lowry [13] and

Brady et al. [2]). An advantage of this kind of description has been pointed out by Connell [5] in terms of "functional improvisation". An example will make the concept clear. Suppose you want to hit a nail and there is no hammer around. You are likely not to give up but to use something else as hammer, say a screwdriver or a shoe heel. This means humans have the capacity to use things for functions different from their main design goal. The shape-function connection is therefore more complex than a simple one-to-one relation between actions and objects, e.g. hit-hammer, sit-chair and so on. Human can deduce from their object models that the handle of a screwdriver can act as head of a hammer if needed.

Ingrand and Latombe [12] have developed a system, SERF (Expert System for Functional Reasoning), where complex structures are deduced from functional constraints, that is a set properties related to the object function. Complex structures are generated composing simpler structures and functional constraints propagate through resulting levels. This work has been inspired by the work of Newell and Freeman [11] where objects are classified according to their function. Composing parts to get complex structures is possible if a part perform the function that another part requires. Two strategies are indicated for parts composing: forward chaining and backwards chaining. The former considers how to build a structure fulfilling a required function starting from objects; the latter how to build an assembly of objects starting from the required function.

Moreover, functional descriptors of objects introduce an explicit link with actions and therefore with the plans involving the objects through the actions [2,3]. The problem of object modeling becomes the relation between their structural properties and their function, which is in turn related to the actions they make possible. Winston et al. [19] have suggested that functional descriptors could be learnt in terms of structural features, but the consequences of this idea hasn't been implemented so far. They point out another substantial advantage of the functional approach, ,it is often too hard to tell vision systems what things look like; it can be easier to talk about purpose and link purpose to functional constraints. This means that the unique descriptor of an object class can be used to recognize instances which look even very different from one another.

One part of the "Mechanic's Mate" project is the study of how function can be deducted from shape and vice versa. In [2], Brady et al. introduce a list of geometric features of common tools as related to the function they accomplish in the tool. The first aim of the project is to understand the interplay between planning and reasoning that involves tools and shape representation for those parts. This as been called by Lowry "reasoning between structure and function" [13]. The structure-function relation is closely dependent on the dynamic representation of the world, which can be given in terms of naive physics [10,6]. The basic idea is that it is often difficult to separate the function of a tool from the plan in which it participates and with which it coevolved. Plans and the tools that they employ coevolve and differentiate with time. Knowledge of the function can therefore make the recognition process easier.

2. SEMANTIC NETWORKS AS FUNCTIONAL DESCRIPTORS

In the analysis of many common objects classes some design goals come repeatedly: supporting, containment, graspability and so forth. Such functionalities describe expected object performances and can be associated to a set of so-called **functional primitives** [8,9,7]. Many objects can be characterized referring to this basic set of functionalities and recognized combining structural and functional information. Deducing function from shape involves some kind of **structural decomposition** which must be coupled with functional information about the subparts. Broadly speaking, a chair can be described as "something you can sit on without falling back", which translates in two main functional primitives, SUPPORT and BLOCK-ROTATION. At a lower level, they can in turn be described in terms of geometric constraints which the chair subparts must satisfy.

The relation between subparts geometry and their functionality has been described by means of semantic graphs commonly used for natural language [18], with minor modifications. The following semantic cases are used: **actor,** the entity which performs the action; **object,** the entity which undergoes the action; **directive,** the axis along which the action is performed; **instrument,** the entity used to perform the action.

The relation between structure and functionality is hidden in the functional primitives. Sometimes, however, functional specification of some structural details would require a huge amount of physical knowledge, or it might be the case that a function imposes loose constraints on structure (e.g., you can HIT a nail with objects of very different shapes). A second set of primitives has therefore been defined, called **spatial primitives.** They are concerned with geometric and spatial relationships. The need for primitives instead of simple labelle arcs is due to the complexity of the spatial knowledge. For instance, the concept of OVER implies the capability of evaluating the relative positions of two surfaces, considering equilibrium if needed and identify which parts can accomplish all this.

The whole knowledge about an object is therefore embedded in a **semantic functional descriptor,** that is a semantic network whose functional primitives share some fillers to express how subparts concur to perform complex functions. Such a network can be interpreted to carry out recognition process. Thiss will be analysed in detail in the next paragraphs. A complete example of semantic functional descriptor for a chair is given in figure 1.

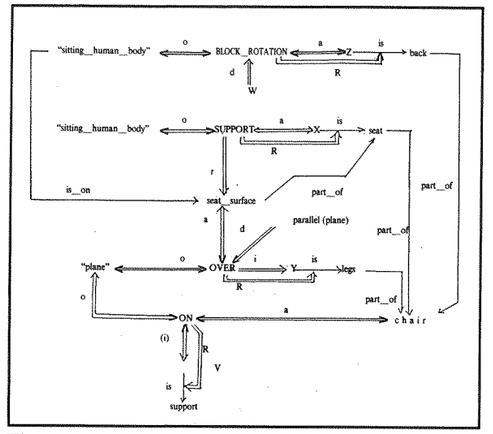


Fig. 1 The chair

3. DISTRIBUTING FUNCTIONAL KNOWLEDGE IN AN ACTOR NETWORK

The formal scheme sketched above is being implemented in an actor-based system. The actors are divided in two classes: **expert-actors** (EXA's) and **ego-actors** (EGA's).

The EXA's correspond to the functional primitives in the semantic descriptor. They know about themodalities of the particular action related. EXA's in a semantic descriptor don't know about the existence of other EXA's: they are not directly connected to one another nor can they directly communicate. They exchange information only with their close neighbours, that's actors of the second class (see below). Therefore the EXA's internal structure is completely independent of the semantic descriptor to which it belongs. In the recognition process, EXA's main activity is to find possible entities capable of exploiting a given function. This search takes place on a 3-D description of the scene as it could be generated by an integrated early vision system [14, 15] under development at our Department. This activity generates a number of candidates or **local hypotheses** each of which is formed by two main components: a **pointer** to some parts of the scene description, and a **local uncertainty coefficient** quantifying the local consistency of the hypothesis. this will be used later during the aggregation of local hypotheses into compound structures hypotheses, which is the task of the second class of actors.

EGA's correspond to the semantic roles in semantic descriptors. They are directly connected to EGA's and are responsible of the formation and evaluation of compound hypotheses from local ones. EGA's describe the subparts of the object represented by the semantic descriptor. They evaluate also the uncertainty of compound hypotheses using local uncertainty coefficients. In other words, they receive local hypotheses from EXA's (possiby even from neighbouring EGA's) referring to they their subparts, aggregate them in a compound hypothesis and evaluate its consistency. Such process stretches throughout the network creating more and more complex entities until the so-called **target EGA** is reached, that is the EGA referring to overall object.

Following the example given in fig. 1, a "chair" hypothesis will be formed starting from the formulation and evaluation of local hypotheses for seat, back, legs, provided by the appropriate EXA's (SUPPORT, BLOCK-ROTATION, OVER); then across the compound group seat-legs up to the formulation and evaluation of the complete chair structure by the target EGA.

4. MORE ABOUT HYPOTHESES: DETAIL AND REFUSAL

The internal activity of an EXA runs in two steps: **prerequisite analysis** (PA) and **detail analysis** (DA). The very first action of an EXA during the recognition process is an attempt to locate some parts of the scene description which satify some essential functional requirements. This makes use of a resctricted number of constraints, generating a first group of entities each with a local uncertainty coefficient. Such entities will not be necessarily part of the recognized object, as their acceptance/refusal as part such can take place only after a global analysis. The PA enables the system to drop with little effort unsuitable candidates and at the same time to reveal objects which cannot definitely be labeled as, say, tables, but could indeed be used as tables (e.g. a bank), thus providing a sort of "functional improvisation".

The PA comes short at least in two cases. First, the target EGA cannot definitely accept a global hypothesis whose substructures have been all generated in PA state. To obtain a more detailed evidence is the task of the DA. For instance, the PA of the SUPPORT EXA will search for seat surfaces simply defined as more or less planar surfaces of a certain size, no matter if the surfaces is continuous or composed by, say, small parallel rods. The DA is activated only for those EXA's which describe the **primary function** of the semantic descriptor, e.g. SUPPOT for the chair, CONTAIN for a jug and so on.

The second case where a simple PA analysis is insufficient is that of an EGA rejecting a hypothesis after evaluating it. In this case a backtrack is started and the EXA's involved must go beyond the PA.

5. PARALLELISM, COMMUNICATION AND SCHEDULING

Different subparts of a single network describing a compound object could be active simultaneously and cooperate exchanging hypotheses. According to the actor paradigm, EXA-EGA actors coordinate their activities exchanging messages. Both the communication and the virtual parallelism are simulated in the system by means of a **scheduler** whose task is threefold: a) it divides the time into **activity slots,** where a group of actors can work

"in parallel". The slot is defined as the interval of time needed to serve all the actors in the queue; b) it controls the message exchange exploring a **mailbox,** where actors leave their messages for other actors. Each message contains a sender address, a receiver address, a priority and an information field; c) it organizes the activity group composed by actors which can execute in the same slot. Exploring the state of the mailbox at the end of each slot the scheduler creates an **activation queue** which encodes the sequence of the actors to be activated in the next slot.

6. DISTRIBUTED BACKTRACKING AND FAIL MESSAGES

When an EGA rejects a compound hypethesis another one must be formulated if possible. It might be simply the case that some functional substructures need detailed investigation, or some candidates might be quite wrong. In both cases the rejecting EGA starts a backtracking distributed over the set of its collaborators. It must therefore know which of them is to revise first. This information is stored in tree structures where cooperating actors are organized hyerarchically. The deepest level in the tree comprehends those actors responsible of the simplest substructures known by the EGA. These actors will be reactivated first.

The reactivation is accomplished by means of **fail messages.** They are high-priority messages that the rejecting EGA places in the mailbox. Their effect is that the receiver will be scheduled before any other actor in the activation queue. This enables to brush from the queue actors whose action has become meaningless upon dismantling the hypothesis. An essential scheduler cycle can therefore be written as:

```
scheduler_cycle()
   create_queue_from(mailbox);
while(queue_not_empty) do begin
   serve(first_in_queue);
   if (interr_in_mailbox) then acktr_cycle();
end
end.
```

A first system prototype has been implemented on a DEC GPX Workstation running ULTRIX. The scheduler organizes the activity of a network related to a CHAIR functional semantic descriptor. Input data are a volumetric representation of a quasi-standard chair which makes the SUPPORT search generate several hypotheses: since the chair might be recognized in any position with respect to the ground, all the plane surfaces are considered possible seat surfaces (PA phase). The network rejects actually the wrong seat surface hypotheses for the structural constraints due to the connections among experts, accepting eventually the input data as a possible chair (DA phase). A graphic interface monitors the analysis running showing the status of the mailbox, the activation queue, the eperts active in each slot and the data base structures under analysis.

7. CONCLUSION AND FUTURE TRENDS

Functional descriptors have been introduced as model for avoiding the impasse of shape-based recognition. A set of functional primitives and their structures as elements of semantic networks have been proposed. A parallel implementation based on the actor paradigm has been discussed. Ongoing work is aiming to refine the set of primitives currently used and to improve the description of the shape-function relationship. Semantic functional descriptors should enable not only to recognize wide classes of objects but also to reason about objects functionalities.

8. REFERENCES

[1] P. Besl, R. Jain, THREE-DIMENSIONAL OBJECT RECOGNITION, Compunting Surveys XVII(1), 1985

[2] M. Brady, P. Agre, D. Braunegg, J. Connell, THE MECHANIC'S MATE, ECAI 84, Advanced in Artificial Intelligence, Elsevier Science Publisher B.V., North-Holland, 1984

[3] M. Brady, ARTIFICIAL INTELLIGENCE AND ROBOTICS, Artificial Intelligence XXVI pp. 79-121, 1985

[4] R. Brooks, SYMBOLIC REASONING AMONG 3-D MODELS AND 2-D IMAGES, Artificial Intelligence WVII, 1981

[5] J. Connell, LEARNING SHAPE DESCRIPTIONS: GENERATING AND GENERALIZING MODELS OF VISUAL OBJECTS, MIT Department of Electrical Engineering and Computer Science, M.S. Thesis, 1985

[6] K. DeKleer, QUALITATIVE AND QUANTITATIVE KNOWLEDGE IN CLASSICAL MECHANICS, MIT AI Laboratory, AI-TR-352, 1975

[7] M. DiManzo. G. Adorni, F. Ricci, F. Giunchiglia, BUILDING FUNCTIONAL DESCRIPTIONS, Proceeding 5th ROVISEC, October 1985

[8] M. DiManzo, G. Adorni, F. Ricci, A. Batistoni, C. Ferrari, QUALITATIVE THEORIES FOR FUNCTIONAL DESCRIPTION OF OBJECTS, Esprit Report P419-TK4-WP2-D11, May 1986

[9] M. DiManzo, F. Ricci, A. Batistoni, C. Ferrari, A FRAMEWORK FOR OBJECT FUNCTIONAL DESCRIPTIONS, Proc. 2nd Intern. Conf. on Artif. Intell., December 1986

[10] K. Forbus, QUALITATIVE PROCESS THEORY, MIT AI Laboratory, AIM-664A, 1983

[11] P. Freeman, A. Newell, A MODEL FOR FUNCTIONAL REASONING IN DESIGN, Proc. IJCAI, 1971

[12] F. Ingrand, J.C. Latombe, FUNCTIONAL REASONING FOR AUTOMATIC FIXTURE DESIGN, Proc. 13th Annual Meeting "Man or Machine: A Choice of Intelligence", 1984

[13] M. Lowry, REASONING BETWEEN STRUCTURE AND FUNCTION, Proc. Image Understanding Workshop, Baumann, Science Application Inc., 1982

[14] M. Maresca, P. Morasso, V. Tagliasco, T. Vernazza, MEMORIA TRIDIMENSIONALE PER LA RAPPRESENTAZIONE DI OGGETTI (in italian), AICOGRAPHICS-85, 4-8 November 1985

[15] P. Morasso, G. Sandini, 3D RECONSTRUCTION FROM MULTIPLE STEREO VIEW, Third Intern. Conf. "Image Analysis and Processing", Intern. Assoc. for Pattern Recognition, October 1985

[16] R. Nevatia, T. Bindford, DESCRIPTION AND RECOGNITION OF CURVED OBJECTS, Artificial Intelligence VIII(1), 1977

[17] A. Requicha, REPRESENTATION OF RIGID SOLIDS: THEORY, METHODS AND SYSTEMS, Computing Surveys XII(4), 1980

[18] R. Schank, CONCEPTUAL INFORMATION PROCESSING, North-Holland, Amsterdam, 1975

[19] P. Winston, T. Bindford, B. Katz, M. Lowry, LEARNING PHYICAL DESCRIPTIONS FROM FUNCTIONAL DEFINITIONS, EXAMPLE, AND PRECEDENTS, Robotics Research, pp. 117-135, MIT Press, 1984

FISICA NAÏVE: UN SISTEMA PER LA SIMULAZIONE E L'INTERPRETAZIONE DEL COMPORTAMENTO DEI LIQUIDI

Hadley Taylor (*), Paolo Stofella (*)

RIASSUNTO

Si descrive nel seguito la realizzazione di un sistema per la simulazione ed interpretazione del comportamento dei liquidi. La simulazione è basata sull'utilizzo di una rappresentazione analogica, sulla scomposizione molecolare delle sostanze fisiche, ed è realizzata mediante la definizione di regole intuitive per la descrizione del comportamento delle sostanze fisiche.
Il sistema di simulazione è stato interfacciato con un sistema di interpretazione che fornisce un insieme di conoscenze sullo stato dell'ambiente fisico rappresentato; in particolare esso rende disponibile ad una sistema di ragionamento informazioni sulle relazioni tra il liquido e gli oggeti, utili per gestire una interazione intelligente tra un agente esterno e l'ambiente di simulazione.

ABSTRACT

A description of a system for the simulation of the physical behavior of liquids is given togheter with the description of a separate component which can interpret and formalize specific knowledge on the environment of the simulation. The simulator is based on an analogical representation, on a molecular decomposition of physical substances and is based on a small set of intuitive rules which describe the behavior of the physical substances.
The interpreter described can analyze the simulation environment to supply qualitative information on the relations between the liquid and the objects, to a reasoning system. This information is eventually used by a reasoning system to guide an intelligent interaction with the simulation environment.

1. INTRODUZIONE

Nell'ambito delle ricerche legate alla disciplina della Intelligenza Artificiale è presente un crescente interesse per la modellazione del ragionamento qualitativo e di tipo "common sense" relativo al mondo fisico.

Quest'area di ricerca, che prende il nome di Fisica Naïve [4,5], si propone di mettere a punto formalismi e modelli computazionali in grado di operare inferenze relative al mondo fisico anche in casi nei quali l'insufficienza di descrizioni quantitative complete di sistemi e processi fisici o dei parametri coinvolti non consente l'utilizzo dei modelli sviluppati nell'ambito della fisica classica. La Fisica Naïve può offrire spunti e modelli utili in settori dell'Intelligenza Artificiale quali quelli per la costruzione di sistemi di controllo per robot o per lo studio dei sistemi esperti.

Il lavoro presentato nel seguito si inserisce nell'ambito delle ricerche sul ragionamento qualitativo privilegiando, a differenza di molti dei lavori descritti in letteratura [2,6,3], quegli aspetti dell'attività mentale che sono legati alla manipolazione di informazioni in forma pittorica del mondo fisico.

Obiettivo della ricerca è quello di fornire uno strumento per la simulazione e interpretazione del comportamento dei liquidi basato sulla conoscenza intuitiva del mondo e delle sostanze fisiche.

2. SIMULAZIONE

Il sistema per la simulazione del comportamento dei liquidi è stato costruito seguendo due direttive fondamentali:

(*) Attualmente presso la Artificial Intelligence Software S.r.l. (Milano).

a) l'utilizzo della rappresentazione analogica (diretta) per l'ambiente fisico che è oggetto della simulazione [8,9] [1]

b) la suddivisione delle sostanze fisiche in unità elementari che saranno chiamate molecole (senza riferimento all'accezione chimico-fisica del termine). Si farà riferimento a tale pratica con il termine di approccio molecolare.

3. RAPPRESENTAZIONE ANALOGICA

Si dice analogica, secondo la definizione di Sloman [8], una rappresentazione nella quale vi sia profonda similarità strutturale tra l'oggetto della rappresentazione e la rappresentazione stessa. Esempi di rappresentazioni analogiche sono mappe, carte ed immagini. Ove tale similarità sia assente si fa riferimento a rappresentazioni di tipo fregeano; esempi di rappresentazioni fregeane sono i linguaggi della logica o della matematica, ed anche i formalismi sviluppati nell'ambito della fisica naive per il trattamento di informazioni qualitative [2,6,3].

Rispetto a quest'ultimo insieme di formalismi, le rappresentazioni analogiche presentano vantaggi, soprattutto in termini di minore complessità, nel trattamento di informazioni di tipo topologico e spaziale. Questo fatto ricopre notevole importanza nel trattamento di informazioni sul mondo fisico, in particolare, dunque, nell'ambito della fisica naive [11,10].

4. APPROCCIO MOLECOLARE

Nell'approccio molecolare, ogni sostanza fisica è concepita come un insieme di unità chiamate molecole. Tali molecole hanno percezione ed effetti locali (sono in grado di comunicare solamente con le molecole adiacenti). Sono presenti due tipi di molecole nel sistema: molecole di solido e di liquido. Per tali molecole è stato definito un insieme di regole che ne definiscono il comportamento. Insiemi di regole individuano proprietà intuitive che caratterizzano ii comportamento globale delle sostanze. Per i liquidi si può fare riferimento a proprietà quali la non compenetrabilità molecolare o la fluidità. Questo approccio ha portato ad una notevole diminuzione di complessità nella descrizione del comportamento dei liquidi, rispetto a quella che presumibilmente risulterebbe dall'utilizzo di un approccio non molecolare (si veda, per una formalizzazione logica e non molecolare del comportamento dei liquidi, [7]).

5. REGOLE DI COMPORTAMENTO PER I LIQUIDI

Le regole di comportamento per le molecole di liquido sono state formulate mediante l'utilizzo di osservazioni non tecniche, intuitive, del comportamento globale dei liquidi. L'utilizzo di una rappresentazione analogica è valso a semplificare la formulazione di queste regole: si noti, infatti, il frequente riferimento a relazioni topologiche tra gli elementi del sistema.

Nel seguito si farà riferimento a molecole libere e vincolate; la seguente definizione chiarisce il significato di questa distinzione. Una molecola si dice vincolata quando lo spazio immediatamente sottostante la sua posizione è occupato da un corpo solido o da una molecola vincolata: in caso contrario essa è detta libera.

L'insieme delle regole di comportamento per le molecole di liquido è il seguente:

1) Una molecola può cambiare posizione, occupandone una adiacente, se e solo se questa non è occupata da un'altra molecola o da un corpo rigido (non compenetrazione dei corpi).

2) Una molecola può ricevere messaggi e trasmetterne agli elementi adiacenti (causalità).

3) Una molecola libera continua ad andare verso il basso fino a quando non incontra un corpo rigido o una molecola vincolata; in entrambi i casi essa diviene vincolata (gravità).

4) Una molecola vincolata può muovere in una qualsiasi direzione solo se questo è richiesto da uno degli elementi adiacenti (fluidità).

5) Una molecola vincolata diviene libera se e solo se lo spazio sotto essa è occupato da una molecola libera o diviene libero (fluidità).

6) Se una molecola libera, durante la sua caduta, incontra una molecola vincolata, essa lancia a questa un messaggio di richiesta di spazio (fluidità).

7) Una molecola vincolata che riceva una richiesta di spazio tenta di occupare le posizioni vicine e, quando non è possibile, espande il messaggio alle molecole adiacenti (fluidità).

8) Non può essere occupata una posizione, attraverso l'espansione di richieste di messaggi, che sia più alta della posizione dell'ultima molecola libera che ha espanso il messaggio (fluidità).

6. STRUTTURA DEL SISTEMA

Il sistema di simulazione del comportamento dei liquidi è stato realizzato presso il Centro Comune di Ricerca della Commissione delle Comunità Europee di Ispra, A.I. & Robotics Laboratory, utilizzando il linguaggio Zetalisp ed il Flavor System su una Symbolics Lisp Machine 3600.

Il sistema si compone delle seguenti entità (come descritto in figura 1):

— Un insieme di Flavors che definiscono gli "attori" del sistema (molecole e dispositivi fisici)

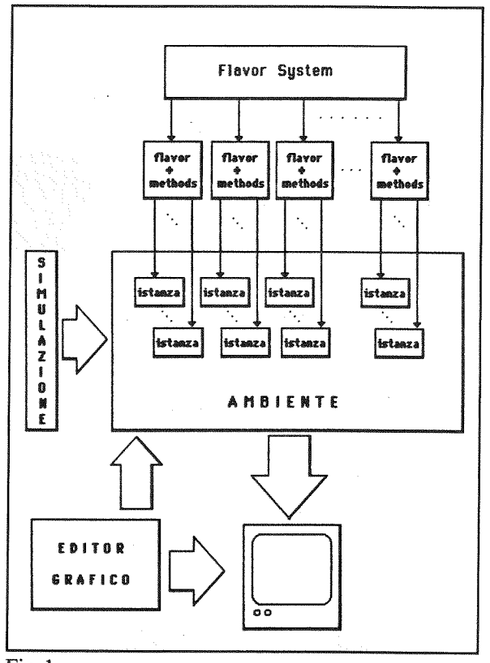


Fig. 1

— Un insieme di Methods; definiscono le regole di comportamento per gli attori

— Una procedura di simulazione; attiva le istanze delle molecole e degli altri attori del sistema

— Un insieme di procedure di uscita grafica che visualizzano l'evoluzione dell'ambiente di simulazione

— Un insieme di procedure di interazione con l'utente del sistema che permettono allo stesso di creare o modificare l'ambiente di simulazione.

7. IL PROCESSO DI SIMULAZIONE

Il processo di simulazione attiva le molecole di liquido e gli altri attori presenti nel sistema fisico, le modalità di comportamenti di questi essendo definite negli insiemi di regole. Ad ogni passo questi attori applicano le regole che li caratterizzano, modificando lo stato dell'ambiente. Più formalmente si ha:

Data una successione di punti temporali

$$t_1, \ldots, t_i, \ldots$$

ciascuno corrispondente ad un passo di simulazione e dati gli attori

$$\{a_1, \ldots, a_n\}$$

sia $S(a_k,t_i)$ lo stato dell'attore a_k al tempo t_i (si intende che lo stato di un attore sia l'insieme dei valori assunti dalla *instance variables* che lo caratterizzano). Allora lo stato dell'ambiente di simulazione è dato da:

$$S(t_i) = \{S(a_1,t_i), \ldots, S(a_n,t_i)\}$$

e vale la seguente relazione:

$$S(a_k,t_{i+1}) = \begin{cases} \text{se } a_k \text{ è attivo} & \text{action } (a_k,S(a_k,t_i)) \\ \text{se } a_k \text{ riceve n messaggi} & \text{action}^n\ (a_k,S(a_k,t_i)) \\ \text{altrimenti} & S(a_k,t_i) \end{cases}$$

ove la funzione action corrisponde all'applicazione delle regole di comportamento.

8. RISULTATI DELLA SIMULAZIONE

Il sistema di simulazione che abbiamo descritto, fornisce un comportamento intuitivamente corretto in una grande varietà di situazioni, comprendenti riempimento e svuotamento di contenitori di forma qualsiasi e funzionamento di circuiti idraulici. È da notare, comunque, che tale sistema si occupa solamente dell'evoluzione dei sistemi fisici da un punto di vista cinematico.

9. IL SISTEMA INTERPRETER

Il Sistema Interpreter per la simulazione del comportamento dei liquidi descritto qui di seguito, mette in rilievo come si possa realizzare la connessione fra la formalizzazione della conoscenza "common sense" e l'utilizzo di immagini in un modello di ragionamento computazionale [10].

È necessario tenere presente che la simulazione analogica dei liquidi, per la quale è stato sviluppato questo sistema di interpretazione, è basata non solo su una visualizzazione bidimensionale del liquido e degli oggetti solidi, ma anche su una rappresentazione interna bidimensionale.

L'Interpreter fornisce informazioni sul liquido di cui viene simulato il comportamento sempre in relazione agli oggetti con cui esso interagisce. In altri termini analizza gli oggetti e ne fornisce una descrizione in cui compaiono informazioni sulla loro situazione rispetto alla loro relazione con il liquido. Potremmo dire che l'interprete è object-oriented, in contrapposizione a liquid-oriented. La motivazione è dovuta al fatto che, avendo il liquido proprietà molto variabili (cf. Hayes 5), non si ragiona in genere in termini di esso, ma in termini dell'oggetto(i) che ne determina la proprietà.

Ad esempio, in riferimento alla situazione in figura 2 l'Interpreter non fornirà una descrizione esplicita per il liquido del tipo "il liquido è contenuto dall'oggetto", ma fornirà una descrizione del tipo "l'oggetto contiene del liquido".

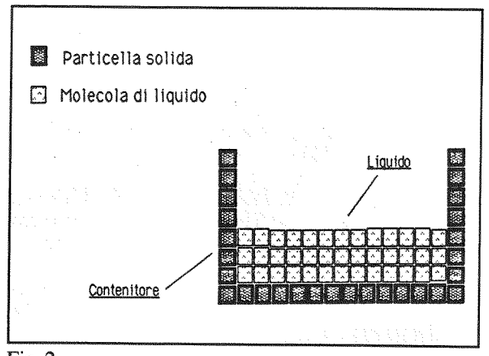


Fig. 2

Il Sistema Interpreter è object-oriented anche nel senso dello stile di programmazione usato per implementarlo: fa uso di "flavors".

Per ogni oggetto che l'Interpreter riconosce viene creata una istanza di "flavor" che lo descrive. Chiameremo questa istanza "descrittore". Le informazioni incluse nel descrittore sono di due tipi: informazioni su liquido che è in qualche relazione (fisica) con l'oggetto e informazioni topologiche e geometriche sull'oggetto.

Le variabili che descrivono informazioni sul liquido sono le seguenti:

CONTAINS: Vale **t** (valore "vero" in Lisp) se l'oggetto contiene liquido, **nil** (valore "falso" in Lisp) altrimenti. In termini implementativi un oggetto contiene del liquido se all'interno di una sua concavità (per definizione di concavità vd. sotto) vi è almeno una molecola vincolata che nella posizione sottostante ha una particella dell'oggetto.

PROB-FULL: Vale **t** se almeno una concavità dell'oggetto è completamente riempita dalle molecole di liquido, **nil** altrimenti.

FILLING: Vale **t** se in un dato istante si sta riempiendo almeno una concavità dell'oggetto.

FLOW-ON-SURFACE: Vale **t** se in un dato istante vi è del liquido che sta fluendo su una superficie verticale od obliqua dell'oggetto.

FLOWING-FROM: Questa variabile contiene una lista con gli identificatori degli oggetti dai quali proviene liquido che fluisce direttamente verso l'oggetto descritto. La lista può essere ovviamente vuota (valore **nil**).

EXPANDING: Vale **t** se in un dato instante vi è del liquido che si sta espandendo lungo una superficie orizzontale dell'oggetto.

Le principali variabili che descrivono informazioni topologiche e geometriche sull'oggetto sono le seguenti:

FIRST-COORDS: Coordinate di riferimento degli elementi dell'oggetto descritto, utilizzabili per la localizzazione dell'oggetto nell'ambiente fisico simulato.

CONTACTS: Lista di identificatori di oggetti che hanno almeno una particella adiacente ad una particella dell'oggetto descritto. Se un oggetto è in contatto con un altro (che abbia un identificatore distinto), oltre alla descrizione dei singoli oggetti, viene fornita anche una descrizione per l'oggetto composto. In questo modo è possibile riconoscere delle proprietà che risultano dall'accostamento di due (o più) oggetti. Ad es. in figura 3 sono rappresentati due oggetti distinti che nella posizione attuale non possono contenere liquido (secondo la definizione di CONTAINS vista sopra). Se i due oggetti vengono avvicinati fino ad entrare in contatto fra loro (fig. 4), essi, secondo il modello di simulazione bidimenzionale considerato, possono contenere del liquido. In tal caso non sarebbe corretto dire che l'oggetto **"A"** o che l'oggetto **"B"** contengono del liquido, ma sarebbe corretto dire che l'oggetto composto **("A" "B")** contiene del liquido.

CONCAVITIES: Lista di coppie di coordinate di tutte le posizioni nell'**array** interno che corrispondono a punti di concavità dell'oggetto descritto. Le concavità considerate dall'Interprete sono sempre e solo quelle rivolte strettamente verso l'alto. Sono solo queste in fatti che possono contenere del liquido.

PARTIALLY INCLUDED: Lista di identificatori di oggetti almeno parzialmente inclusi in una concavità dell'oggetto descritto.

TOTALLY-INCLUDED: Lista di identificatori di oggetti interamente inclusi in una concavità dell'oggetto descritto.

10. SISTEMA DI RAGIONAMENTO

Utilizzando il sistema di simulazione del comportamento dei liquidi basato sulla rappresentazione analogica ed il

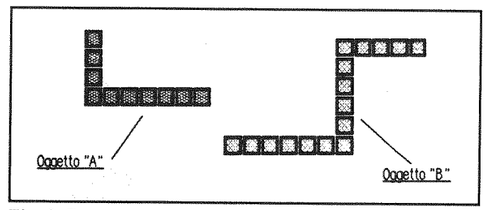


Fig. 3 Contacts

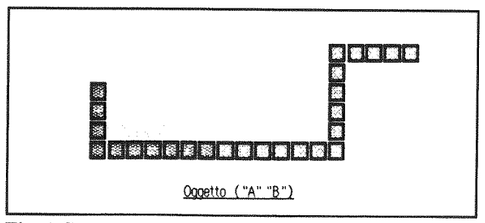


Fig. 4 Contacts

sistema Interpreter, è possibile modellare il sistema agente(uomo-robot)-ambiente fisico, nell'ambito della risoluzione di problemi che coinvolgono conoscenza di tipo common-sense sul comportamento dei liquidi. Chiamiamo questo modello il Reasoning System.

Il nostro primo approccio alla realizzazione del Reasoning System è stato diretto allo sviluppo di un programma, il Pouring Program che utilizza un insieme di euristiche analogiche, specifiche del modello di rappresentazione, basate sui risultati delle inferenze svolte dalla simulazione, e le informazioni raccolte dall'interprete sulla situazione rappresentata, per operare delle azioni sugli oggetti al fine di versare liquido da un contenitore di qualsiasi forma ad un altro.

L'aspetto più interessante del programma è il modo in cui alterna l'attivazione dei diversi componenti del sistema di ragionamento che gestisce: la simulazione, l'interpretazione ed i propri schemi procedurali di ragionamento. È infatti per mezzo di questo ciclo continuo che viene evidenziata la capacità del sistema di utilizzare una conoscenza di tipo common-sense per agire interattivamente con il mondo fisico.

11. CONCLUSIONI

I sistemi che abbiamo descritto tendono a mettere a punto una metodologia che si presta ad essere generalizzata ad un ampio ambito applicativo, all'interno della fisica naive. L'utilizzo di un simulatore per il comportamento di oggetti fisici integrato con un sistema di interpretazione permette di operare sul mondo fisico sfruttando ad un tempo i vantaggi della rappresentazione analogica nel trattamento di informazioni spaziali, la minore complessità derivante dall'approccio molecolare e la disponibilità di informazioni qualitative nell'ambiente fisico fornite dall'interprete. L'utilizzo di questi strumenti in un semplice sistema ragionatore ne mette in evidenza le possibilità d'impiego in settori diversi all'interno dell'Intelligenza Artificiale (nel caso specifico si fa riferimento ai problemi di planning).

12. RINGRAZIAMENTI

Si ringraziano il prof. Bernard Meltzer, il prof. Francesco Gardin, la dott.ssa Stefania Bandini, il prof. Gianni Degli Antoni ed il dott. Flavio Argentesi per i contributi apportati alla realizzazione di questo lavoro.

13. BIBLIOGRAFIA

[1] F. Gardin, P. Stofella, B. Meltzer, THE ANALOGICAL REPRESENTATION OF LIQUIDS IN NAÏVE PHYSICS, Proceedings E.C.A.I.-'86, 1986

[2] J. DeKleer, J.S. Brown, A QUALITATIVE PHYSICS BASED ON CONFLUENCES, Artificial Intelligence vol, 24, 7-84, 1985

[3] K.D. Forbus, QUALITATIVE PROCESS THEORY, Artificial Intelligence vol. 24, 85-169, 1985

[4] P.J. Hayes, THE NAÏVE PHYSICS MANIFESTO, Expert Systems in the micro-electronic age Donald Michie, ed. Edinburgh, Scotland Edinburgh University Press, 242-270, 1979

[5] P.J. Hayes, THE SECOND NAÏVE PHYSICS MANIFESTO, Formal Theories of the commonsense world, ed. J.E. Hobbs and R.C. Moore, 1-36, 1985

[6] B. Kuipers, COMMONSENSE REASONING ABOUT CAUSALITY: DERIVING BEHAVIOR FROM STRUCTURE, Artificial Intelligence, vol. 24, 169-204, 1985

[7] P.J. Hayes, NAÏVE PHYSICS 1: ONTOLOGY FOR LIQUIDS, Formal Theories of the commonsense world, ed. J.E. Hobbs and R.C. Moore, 71-109, 1985

[8] A. Sloman, INTERACTIONS BETWEEN PHILOSOPHY AND ARTIFICIAL INTELLIGENCE: THE ROLE OF INTUITION AND NON LOGICAL REASONING IN INTELLIGENCE, Artificial Intelligence, vol, 2, 209-225, 1971

[9] A. Sloman, WHY WE NEED MANY KNOWLEDGE REPRESENTATION FORMALISMS, British Computer Society Expert Systems Conference, 1984

[10] H.C. Taylor, STUDI SULLA MODELLAZIONE COMPUTAZIONALE NELLA FISICA NAÏVE, Tesi di laurea, Università di Milano, Facoltà di Scienze Matematiche, Fisiche e Naturali, Corso di laurea in Scienze dell'Informazione, 1986

[11] P. Stofella, LA RAPPRESENTAZIONE ANALOGICA DEI LIQUIDI NELLA FISICA NAÏVE, Tesi di laurea, Università di Milano, Facoltà di Scienze Matematiche, Fisiche e Naturali, Corso di laurea in Scienze dell'Informazione, 1986

A FRAME LIKE SYSTEM IN PROLOG FOR REPRESENTING KNOWLEDGE

Stefania Bandini, Fabio Buciol

Dipartimento di Scienze dell'Informazione

Università di Milano

Angelo Beltramini

ME.S.A. - Milano

RIASSUNTO

In questo lavoro viene descritto un sistema basato su frame implementato in Prolog. Il sistema comprende una particolare sintassi basata su frame mediante cui è possibile definire contesti.
Il collegamento tra i vari livelli del frame-system è reso possibile da una rete semantica che ne permette anche la gestione dinamica.
Viene inoltre presentato un algoritmo che supporta la fase di pattern-matching nella consultazione della base della conoscenza.

ABSTRACT

This paper deals with the description of a Prolog system fit for frames managing. The system implements a particular frame syntax through which user can define and assing frames according to his own requirements.
Connecting different frames it's possible to draw a semantic network shaped according to the problem studied.
It's also presented an algorithm to look up the knowledge base made up using frames.

1. INTRODUCTION

Particular studies have been carried out in order to point out the power and the limits of Prolog.

Interest in Prolog performances has deeply increased either at the University either at the industrial environment: declaring hypotesis about the world, using facts and rules, as well as drawing conclusions from them through resolution is quite appealing.

Problems about data structure have been considered: the insufficiency of primitive structures can be overcome by using frame like structures to represent complex information about entities or situations.

This paper deals with the description of a system inteded for managing frames in Prolog; its performances can be summarized into three main parts:

— frame syntax definition,

— frame-manager definitions,

— extension of matching process to frame-like structures: an algorithm.

The system realized was tested representing knowledge about planning in industrial process control through a frame network.

2. FRAMES IN PROLOG

Sharing M. Minsky's outcome about frame theory [3] a frame is a formalism to reproduce human mental schemata for the representation of situations or complex entities.

Referring to Minsky's theory, N.J. Nilsson suggests a frame implementation, the Unit [4] a clause conjunction of predicate calculus: these are structures able to represent aspects and features about entities and situations.

A frame could be considered as a collection of slots: each slot represent a feature of an object or a situation described by the frame itself.

Every slot could be composed by:

— a name to indentify which feature the slot expresses;

— a value, the information about the feature;

— a type (or syntax identifier) declaring the syntax of information to assign the slot (it could be a sentence, a single word or a list).

According to these ideas, a general frame syntax was defined: a frame is a Prolog fact with one argument, a list made up by a sublist (see Figure 1).

Each sublist is a frame slot (a terminal).

Each slot could be made up by sublist too, according to its syntactial features.

The system could manage slot with different syntactical features lists of atoms (words), a sentence (as list of words expressing a concept or an idea), lists of sentences, pointers (referring to other frames), frame identifiers and set-names (useful to define relations with frames) (see Figure 2).

As "frame pattern" we mean the information set fil for defining the frame structure itself.

In order to manage a lot of instances of the same kind of event or object it's more efficient to have only one description of the data structure (see Figure 3).

A pattern includes a list of couples of values defining the slots of a frame: the first argument is the slot-name whereas the second one is the syntactic identifier (see Figure 4).

Syntactic identifiers select the procedural segments which are fit for managing the corresponding slot-values.

Semantic network

Single frames can hardly represent complex and pratically meaningful scenarios.

Pointing out relations between objects and entities represented by frames is often useful.

By properly using system primitives, user can easily organize knowledge about problems in a semantic network: nodes are "object" frames while arcs are implemented through "relation" frames.

The system manages "delineation" frames too: they collect information about the network structure.

Frame-manager

The frame-manager is a collection of procedures designed for managing the frame system.

It's made up by a number of general primitives to create, update and look up the knowledge base.

Following paragraphs offer a short shot over the main system primitives.

Define - It lest the user assert a new pattern in the knowledge base. It asks the user about frame name and type, then it acquires name and syntax identifier for each slot.

Create - In using the correct pattern it asks the user values to assign frame slots.

Eventually a new frame is asserted in the knowledge base.

Save - It saves frames and patterns on secondary memory device.

Update - It's a tool to modify frame slot-values.

Delete - It removes a particular frame from the knowledge base.

Cross - It's useful tool to look up the frame system.

User can jump from network node to another following relations: it can also examine information collected in each frame node.

Matching algorithm

Complex data structures like frames need to extend and generalize matching process to frame networks.

Prolog inference mechanism refers to resolution [5] applied to Horn clauses [6] (clauses with at most a positive literal).

Given a clause with an atom "p" on its left-hand side (on the left of: ":—" symbol) and another clause with an atom "p" on its right-hand side, it is possible to create a new clause in which the left-hand side is the union of the left-hand sides of the two original clauses with "p" deleted, and the right-hand side is the union of the right-hand sides of the two clauses with "p" deleted.

The resolution rule is somewhat more general in thet it can be used even when two atoms are not identical, so long as they can be made to look alike by appropriate istantations of their variables. The process of finding such istantations is called unification [5].

The combination of unification with the resolution rule permits more complicated conclusion to be drawn.

Referring to these principles our system matching algorithm verifies informations expressed by a special frame (goal-frame) looking up the knowledge base.

It crosses the frame network looking for frames having slots which unify goal-frame slots (see Figure 5).

In order to manage list-slots and sentence-slots, unification rules have been modified: the algorithm admits two slots unify when atoms belonging to a slot occur in the other one's expression (sentence or list of pointers) (see Figure 6).

Matching algorithm needs goal-frame ("question" frame) definition; its slots report useful indications to find information in the frame network.

Slot values can be variables, either lists of atoms unifying with sentences, complex structures in general or completed information to select a particular frame within the network (see Figure 7).

Matching algorithm can be split into two main parts: the first one identifies the first frame unifying with goal frame; the second one scans network to look for frames to completely verify the goal.

In order to indentify the first frame, the algorithm scans all patterns on the konwledge base: this way the algorithm identifies those frame-sets (made up by frames sharing the same pattern) which most likely include frames unifying with goal because of their structure (see Figure 8).

So it's possible to reduce the number of frames to be scanned in order to find the first unication.

When the first frame has been found, the algorithm looks for the new one scanning connected frames until a new match is possible.

The algorithm repeats the same search starting from the new frame having slots unifying with the goal.

Passing from a node to another the algorithm describes a search tree scanned according to a depht-first strategy [4]: when a node has no connections with frames unifying with the goal, the algorithm backtracks to previous nodes matching with goal to scan relations skipped before (see Figure 9).

3. AN APPLICATION TO PROCESS CONTROL

System effectiveness was experimented representing knowledge about planning activities for process control systems.

Using frame structure it has been possible to describe part of a chemical plant and its regulation system as regulation nodes ("siti"), sequential regulator ("sequenziatori") and physical plant components.

Relations between entities have made explicit by drawing a semantic network (see Figure 10).

We report some example of patterns and frames referring to the semantic network drawn to describe an industrial plant and its control system. in Figures 11, 12, 13.

4. CONCLUSION

The prototype could be considered a first step towards a system supporting industrial plants managing.

An Expert System can be built over the topological plant description enriched by proper rules of inference.

So we obtain a system fit for advising the operator in plant managing since it can look up the semantic network and interpret cause and effect relations.

It's also possible to collect planning specifications and to define a possible solution using frames and semantic networks.

A documentation, automatically managed and shaped according to the problem represented, is an exhaustive and up-to-date picture of acquired knowledge easy to refer using logical relations.

5. KNOWLEDGEMENTS

The system prototype was developed to meet requirements made concrete at ME.S.A. (Montedison group) which deals with system planning for digital direct control.

We expecially thank M. Maiocchi of Computing Science Departement, University of Milan, who contributed to this study by providing helpful comments and advices.

6. BIBLIOGRAPHY

[1] W.F. Clocksin, C.S. Mellish, PROGRAMMING IN PROLOG, Springer-Verlag, 1981

[2] A. Deliyanni, R. Kowalsky, LOGIS AND SEMANTIC NETWORKS, Communications of the ACM 22/3, C. Montgomery Editor, 1979

[3] M. Minsky, A FRAMEWORK FOR REPRESENTING KNOWLEDGE, in P. Wiston, "The Psycology of Computer Vision", McGraw Hill, New York, 1975

[4] N.J. Nilsson, PRINCIPLES OF ARTIFICIAL INTELLIGENCE, Tioga Publishing Company, 1980

[5] M.R. Genesereth, M.L. Ginsberg, LOGIC PROGRAMMING, Communications of the ACM 28/9, C. Montgomery Editor, 1985

[6] F. Pereira, LOGIC PROGRAMMING, Journal of Automated Reasoning, vol. 1, D. Reidel Publishing Company, 1985

[7] F. Buciol, L'USO DI STRUMENTI DI INTELLIGENZA ARTIFICIALE NELLA DESCRIZIONE DI IMPIANTI INDUSTRIALI E NELLA RELATIVA GESTIONE, Tesi di laurea in Scienze dell'Informazione, Università di Milano, 1985/1986

```
FRAME :: = frame-name-n ([slot-list]).
frame-name ::= atom
n ::= number
slot-list ::= [slot] | [slot], ... [slot]
slot ::= atom | atoms | sentence | sentences | pointer | frame | frames | set-name | set-names

atoms ::= atom | atom, atom, ... atom

sentence ::= [sent]
sentences ::= [sent] | [sent], [sent], ... [sent]

frame ::= nil | frame-name
frames ::= nil | frame-name-n | frame-name-n, ..., frame-name-n

set-name ::= nil | frame-name
set-names ::= nil | frame-name | frame-name, ... frame-name

pointer ::= [atom, nil] | [atom, frame-name-n]| [atom, frame-name-n] ..., [atom, frame-name-n]

sent ::= atom | atom, atom, atom, ..., atom

atom ::= prolog-atom
```

Figure 1 Frame syntax

```
meeting1  ([[100286],                                          slot1
           [milan],                                            slot2
           [[paul-withe, person1],
            [john-smith, person2],                             slot3
            [andrew-braun, nil]],
           [[activities, description],
            [different, strategies, introduction],             slot4
            [choice, of, operating, strategy]]
           ]).
```

Figure 2 "Meeting" frame. As an example we can consider a frame representing meetings of work: it summarize information about meeting date, place, interlocutors and topics discussed.

```
PATTERN ::= pattern ([[frame-name], [type], [slots-descr]]).
frame-name ::= atom
type ::= object | relation | delineation-o | delineation-r | question
slot-descr ::= [description] | [description], .... [description]
decsription ::= name, sintax-id
name ::= atom
sintax-id ::= atom | atoms | sentence | sentences | pointer | frame | frames | set-name | set-names
```

Figure 3 Pattern syntax


```
pattern  ([[meeting], [object],
           [[date, atom],                                      slot1 definition
            [place, atom],                                     slot2 definition
            [interlocutors, pointer],                          slot3 definition
            [topics, sentences]],                              slot4 definition
           ).
```

Figure 4 Pattern defining "meeting" frame. Type "object" declares that corresponding frame will be a node semantic network.

```
Procedure match

begin
    input (Goal)
    scan-pattern (Goal, Cand)
    first-unify (Cand, Goal, Fr, Out)
    if   Out="ok"   then
         push (Fr, Stack)
    else return fail
    while not(verif (Goal) or empty(Stack)) do
    begin
         fr-connected (top(Stack), Fx)
         if   unify (Fx, Goal)   then
              push (Fx, Stack)
         else
              if   no-more-rel (Fx)   then
                   pop (Stack)
    end
    if   verif (goal)   then
         return
    else return fail
end
```

Figure 5 Matching Algorithm

```
                    Slot a1                          Slot a1 definition
[activities, strategy]                               [topics, atoms]
unify with          Slot b1                          Slot b1 definition
[[activities, description],                          [topics, sentences]
 [different, strategies, introductions],
 [choice, of, operating, strategy]]

                    Slot a2                          Slot a2 definition
[paul-white]                                         [interlocutors, atom]
unifly with         Slot b2                          Slot b2 definition
[[paul-withe, person1],                              [interlocutors, pointer]
 [john-smith, person2],
 [andrew-braun, nil]]
```

Figure 6 Extended unification. Slots must be consistent at definition level too: the must share the same slot-name and their syntax identifiers must be compatible (7): "atom" and "atoms" can unify either with "sentences" or "pointer" while "sentences" and "pointer" are not compatible at all.

```
                quest1  ([
date                    [100286]        completed information
place                   [v1]            variable
topics                  [strategies, choice]    list of atoms unifying with sentences
name                    [paul-white]    completed information
profession              [v2]            variable
                        ]).
```

Figure 7 Goal-frame exemple

```
Goal-frame pattern

pattern ([[quest], [question,],
          [[date, atom],
           [place, atom],
           [topics, sentences],
           [name, atom],
           [profession, atom]]
          ]).
```

```
Compatible patterns

pattern ([[meeting], [object],
          [[date, atom],
           [place, atom],
           [interlocutors, pointer],
           [topics, sentences]]
          ]).

pattern ([[person], [object],
          [[name, atom],
           [profession, atom],
           [aspects, sentences]]
          ]).
```

Figure 8 Frames, having slots structurally alike or compatible with goal-frame slots, can most likely match with goal-frame itself.

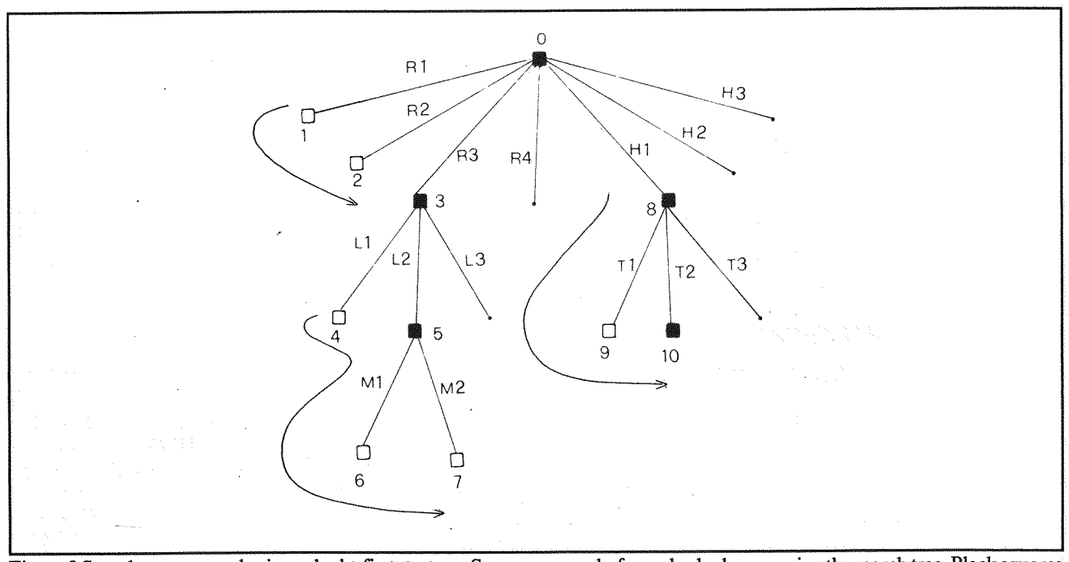


Figure 9 Search tree scanned using a depht-first strategy. Squares are node-frame looked up crossing the search tree. Black squares are frames having slots unifying with goal; figures near frames express the examination sequence: "0" frame is the first frame unifying with goal. Labels and arcs point out relations between frames.

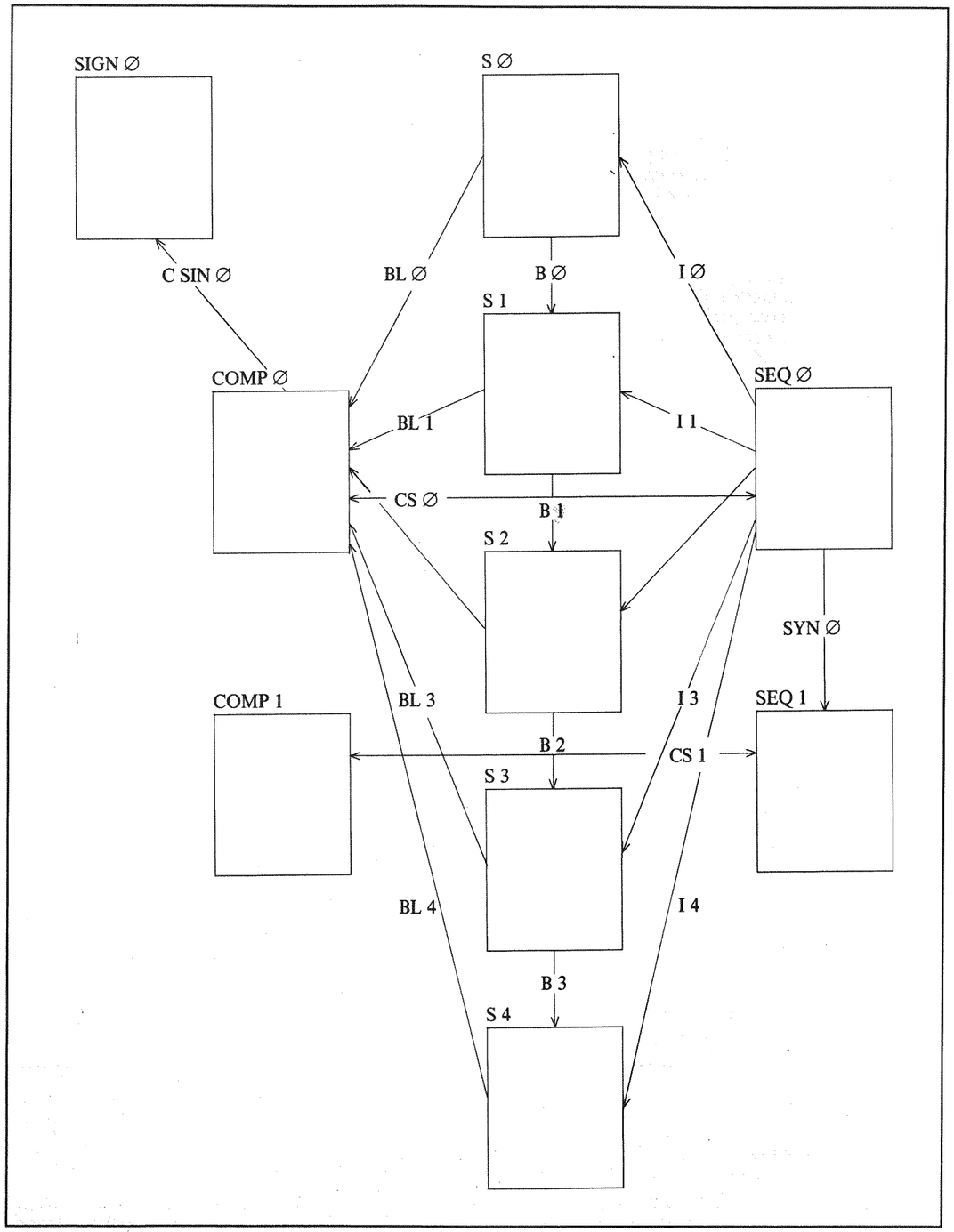


Figure 10 An example of semantic network

```
pattern ([[sito], [object],
          [[sito-name, atom],
           [outputs, atoms],
           [inputs, atoms],
           [preconditions, pointer],
           [conditions-activ, pointer],
           [preconditions-term, pointer],
           [condition-term, pointer]]
         ]).

sito1    ([[styrol-regulation],
           [fts401 (v1-2)],
           [fts401, spfts401, d401-empty],
           [[ok-operator, nil],
            [d401-empty, nil]].
           [[absent, nil]],
           [[set-poin-reached, nil]],
           [[end-of-priming, sito1]]]
         ]).
```

Figure 11 "Sito" pattern and frame. We mean as "sito" (or "regulation" point) that position of plant where regulation happens; each "sito" has a regulation procedure to manage its activity.

```
pattern ([[sequential-regulator], [object],
          [[regulator-id, atom],
           [siti, pointer]]
         ]).

sequential-regulator0([
           [dissolver-activity],
          [[styrol-regulation, sito0],
           [trabsfer-regulation, sito1],
           [nucleante-loading, nil]]]
         ).
```

Figure 12 "Sequential-regulator" pattern and frame. "Siti" regulations are connected by a logical entity: the sequential-regulator. Every sequential-regulator correspond to a control-system task.

```
pattern ([[before], [relation],
          [[previous, frame],
           [next, frame]]
          ]).

before0 ([
           [sito0],
           [sito1]
          ]).
```

Figure 13 A relation. "Siti" regulations occur sequentially: it could be useful to point out the activation order: "before0" relation assert that "sito0" regulation happens before "sito1" one.

UNA ARCHITETTURA KNOWLEDGE BASED PER IL MONITORAGGIO DI AMBIENTI REAL TIME

Emilio Bertolotti

Direzione Sistemi Esperti

Honeywell Information Systems Italia

Pregnana (MI)

RIASSUNTO

Viene presentata l'esperienza di un progetto finalizzato alla definizione e sperimentazione di architetture knowledge based per il monitoraggio in ambienti real time. Si analizzano i problemi generali che emergono nei sistemi inferenziali con base dati in continua evoluzione temporale e vengono descritte le soluzioni trovate in COOKER, un sistema esperto di ausilio agli operatori di un processo industriale.

ABSTRACT

The design and test of knowledge based architectures for real time monitoring is presented. The general problems arising in reasoning with real time data are analized. The solution to these problems, as has been found in COOKER, an advisor system for the operators of a process plant, is given.

1. INTRODUZIONE

Nel 1985 la Honeywell Information Systems Italia ha deciso di partecipare ad un progetto di circa dieci anni uomo, che esplorasse le possibilità nell'ambito "sistemi a base di conoscenza per ambienti real time". Il progetto si è concluso con la realizzazione di due architetture sperimentali e di COOKER, un sistema esperto già installato presso una realtà industriale per la produzione di cellulosa in Pennsylvania-USA, rilasciato in versione Zlisp su Symbolics 3620. La ricerca è stata condotta, oltre che dall'autore, da W. Kaemmerer, J. Allard, J. Nomura presso il Honeywell Computer Science Center di Minneapolis. L'esperienza dello sviluppo di COOKER, un sistema esperto di ausilio agli operatori di controllo di un processo industriale, ha fornito alcuni interessanti spunti che vengono qui discussi. Pur essendo gli argomenti trattati confinati ai sistemi a base di conoscenza per il tempo reale, ed in particolare per il controllo di processo, riteniamo che le problematiche emergenti si prestino ad una attenzione più generale, che va al di là di tali specifici temi. Nella prima parte si presentano le motivazioni del progetto. Nella seconda parte si discutono le problematiche legate alla supervisione di sistemi con un gran numero di sensori in cui le inferenze sono fatte su di una base dati in continua evoluzione temporale. Il modo con cui tali problemi vengono affrontati in COOKER viene descritto nella terza parte.

2. DEFINIZIONE DEL PROBLEMA

La funzione di monitoring di un gran numero di sensori, che permettono di sintetizzare l'astrazione di "stato" di un sistema in continua evoluzione temporale, è oggetto di studio da ormai lungo tempo. Tali sistemi sono in generale descrivibili come una moltitudine di componenti la cui struttura e le cui relazioni sono più o meno ben caratterizzate da modelli fisico-ingegneristici. Possono essere visti sotto questo aspetto sistemi quali un processo industriale, una macchina biologica (uomo), una rete ferroviaria.

L'automazione delle funzioni di monitoring è generalmente ostacolata da alcuni classici problemi legati principalmente alla complessità strutturale del sistema da controllare ed alla complicazione costituita dalla sua rapida evoluzione temporale. È opinione diffusa, ma anche tesi da dimostrare, che con l'avvento dei sistemi a base di conoscenza è possibile infrangere questi ostacoli e quindi potenziare tali funzioni. Il risultato dell'applicazione di questa tecnologia è la creazione di strumenti di ausilio agli

operatori in grado di compiere livelli di sintesi superiori ai sistemi di controllo tradizionali. Mentre il problema del "time dependent reasoning" è ampiamente trattato nella lettura sui Knowledge Based Systems, [1], [2], [3] sono solo alcuni esempi, non molti sono purtroppo i lavori significativi che trattino di sistemi specificatamente orientati al "real time reasoning"; tra quei pochi ricordiamo [5], [6], [7], [8], [9], [10]. Non esiste inoltre, o perlomeno non conosciamo, alcun lavoro che tratti in modo sistematico dei concetti e dei problemi emersi nelle diverse esperienze fino ad ora effettuate.

3. TIPI DI CONOSCENZA RICHIESTA IN UN SISTEMA DI MONITORING

In figura 1 è riportato un esempio di quello che è auspicabilmente richiesto ad un sistema di monitoring di un processo industriale, di alto livello. L'esempio, lungi dall'essere una presentazione esaustiva delle funzionalità desiderabili, permette tuttavia di evidenziare sia i problemi intrinseci delle caratteristiche dei dati su cui inferire, sia il tipo di flessibilità che il meccanismo di problem solving deve possedere. Usando tale esempio come traccia, tenteremo di dare una caratterizzazione dello scenario e del tipo di conoscenza richiesta ad un sistema di monitoring per ambiente real time.

Appare chiaro, fin dalle prime righe, dove sta scritto

"il livello del serbatoio-acido sta diminuendo di 20 litri/minuto"

che è necessaria una conoscenza strutturale, o "profonda" del processo; ossia il processo va rappresentato mediante modelli ingegneristici. È altresì comprensibile come questo insieme di modelli non sia sufficiente, di per se, a catturare tutta la conoscenza che necessità. Per prima cosa, i modelli non coprono la totalità del processo, nel senso che esistono parti non modellabili o che non conviene modellare. Inoltre, esiste una conoscenza di tipo "superficiale", posseduta dagli operatori e frutto della loro esperienza sulle procedure di conduzione del processo, che non è codificabile in modelli ingegneristici, ma è tutt'al più riconducibile ad una serie di criteri euristici.

Generalmente questa conoscenza "superficiale" è quella che permette agli operatori, pur non avvalendosi di alcuna analisi esatta di tipo modellistico, di sintetizzare sensazioni e pianificare azioni che sarebbero, in altro modo, difficilmente ottenibili. Nell'esempio, un tale tipo di conoscenza porterebbe a favorire l'ipotesi del guasto all'indicatore piuttosto che al circuito, una volta noto il buon funzionamento della pompa e della valvola. Una buona descrizione del processo dovrà contenere entrambi i tipi di conoscenza, sia quella di modellistico che quella di tipo euristico.

Quando poi si dice

"cerca di confermare una delle possibili cause basandoti su altre misure, tendenze, aspettazioni"

si deve pensare alla necessità di dover rappresentare le

SE ...

il livello del serbatoio acido sta diminuendo più di 20 litri/min, la pompa dell'acido sta pompando, la valvola è aperta e il flusso di acido segnalato è <10 litri/min

ALLORA ...

potrebbe esserci un problema nell'indicatore di flusso o di livello, o nella pompa, o nella valvola, o nel circuito

QUINDI ...

cerca di confermare una delle possibili cause basandoti su altre misure, tendenze, aspettazioni

DOPO UN TEMPO RAGIONEVOLE ...

se non si è trovata la causa, aumenta il livello di indagine o aspetta qualche evento significativo che certamente accadrà.
Se hai trovato la causa pianifica i rimedi possibili eventualmente chiedendo all'operatore cosa ne pensa, se l'urgenza del problema lo richiede. Se hai più ipotesi cerca di eliminarle tutte meno una

INTANTO ...

continua a verificare che il problema sia presente negli stessi termini, se no, ritratta le precedenti conclusioni

ALLA FINE ...

trai comunque una conclusione e rispondi alle eventuali domande dell'operatore sulla tua conclusione.

Fig. 1 Esempio di monitoraggio ad alto livello

relazioni causali tra le diverse componenti del processo, siano esse contenute nei modelli euristici, in modo completo. Ciò vuol dire che, qualunque sia il livello di granularità scelto nella descrizione dei sottosistemi in cui è scomponibile il processo (dai principi primi della fisica su fino ai maggiori livelli di sintesi), è comunque necessario garantire, oltre che una copertura totale delle relazioni causa-effetto, anche una loro rappresentazione temporalmente estesa, nel passato (tendenze) che nel futuro (aspettazioni). La completezza nella rappresentazione delle relazioni causali deve essere quindi sia funzionale che temporale.

Tornando all'esempio, se si esamina il significato dell'affermazione

"continua a verificare che il problema sia presente negli stessi termini, se no, ritratta precedenti conclusioni"

se ne evince la necessità del sistema di inferire su di una base dati (sensori) congruente e il più aggiornata possibile. La situazione ideale sarebbe quella in cui tutti i dati sono riferiti allo stesso istante di tempo, ciò che garantisce la congruenza, ed ogni aggiornamento viene istantaneamente percepito dal sistema. In realtà, non solo il ritardo nell'aggiornamento è fisicamente ineliminabile, ma addirittura quanto più si tenta di tenere basso questo indesiderato ritardo, tanto più si alza la propensione all'inconsistenza temporale della base dati. Questo conflitto tra le due proprietà desiderate è dovuto al fatto che abbreviando il ciclo di aggiornamento dei dati, rendendolo asincrono rispetto alle aspettazioni che il sistema si forma sul processo, si ha una sensibilizzazione a configurazioni di dati spurie dovute a situazioni transitorie. Di questo ineliminabile problema ne deve tener conto la strategia di problem solving ricercando il giusto compromesso tra le due contrastanti necessità.

Un altro scoglio, questa volta più generalmente comune a molti knowledge based systems e che ricorre anche in questo dominio, è quello dell'incompletezza e ambiguità dell'input. Si pensi infatti alla necessità di dover considerare la lettura dello stato di un sensore come un input che può essere inficiato da malfunzionamenti del sensore stesso.

4. COOKER: LA REALIZZAZIONE DI UNA ARCHITETTURA KNOWLEDGE BASED PER IL REAL TIME MONITORING

Da quanto detto, risulta evidente il tipo di flessibilità richiesta ad un sistema di monitoring del tipo descritto dall'esempio di fig. 1. Mostreremo ora come i problemi suddetti siano stati affrontati in COOKER.

4.1 Rappresentazione della conoscenza ingegneristica ed euristica in una schema integrato

Lo schema di rappresentazione scelto in COOKER è quello di una rete (lattice) gerarchica di oggetti o GOALS. Un GOAL è un obiettivo, per raggiunre il quale, vanno soddisfatti i suoi GOALS figli, ossia i suoi SUBGOALS. Ogni SUBGOAL ha a sua volta dei figli da soddisfare. La catena prosegue fino ai livelli più bassi della rete in cui trovano posto i GOALS "atomici" non ulteriormente scomponibili. La struttura, che almeno a prima vista, assomiglia a ciò che comunemente si definisce un "AND/OR-tree", vedremo (punto 3.3) che in realtà è qualche cosa di completamente diverso. In figura 2 è riportato un frammento semplificato della rete di GOALS in COOKER. Il raggiungimento del top-level goal che è quello di cuocere la cellulosa (COOK), è condizionato dal raggiugimento dei SUBGOALS di livello immediatamente sottostante rappresentanti le diverse fasi del processo (Chip-load, acid-charge ...). A loro volta ogni fase si compone di SUBGOALS, ossia di sottofasi.

Ai livelli più bassi della gerarchia i GOALS sono obiettivi elementari quali l'apertura di una valvola o l'avviamento di una pompa. Ogni GOAL è un oggetto con diversi attributi che intervengono nell'attività di problem solving, ad esempio, in qualità di condizioni da rispettare per il soddisfacimento, o di requisiti da validare prima di tentare il soddisfacimento. L'inference engine è l'animatore di questa rete che visita in ogni suo nodo, guidato dalla sua strategia globale e dalle informazioni locali contenute negli attributi dei GOALS. Diamo per completezza in figura 3 l'elenco dei principali attributi ascrivibili a ciascun GOAL, mentre tratteremo della loro interpretazione nei punti 4.2 e 4.3.

Preferiamo a questo punto fare qualche considerazione generale sullo schema di rappresentazione della conoscenza adottato. Gli attributi di ciascun GOAL sono i depositari della conoscenza; contengono ad esempio le condizioni che decretano il successo o che individuano l'insorgere di un evento indesiderato. In essi sono omogeneamente espresse entrambi i due tipi di conoscenza, ingegneristica ed euristica, o profonda e superficiale se si preferisce, di cui si è trattato al punto 3. Infatti, per esempio, i CRITERI DI SUCCESSO (fig. 3) di un dato GOAL, ossia quei criteri verificati i quali è possibile dichiarare il GOAL soddisfatto, possono dipendere sia dall'esito di una simulazione innescata per un dato modello ingegneristico, sia da condizioni che emulano le aspettative dell'operatore in quel dato contesto. Si ha così un unico schema di rappresentazione dei due diversi tipi di conoscenza che, se si differenziano nelle diverse primitive che utilizzano, convivono e sono esprimibili, ad un più alto livello, in modo del tutto analogo.

Un ulteriore elemento di omogeneità deriva dalla possibilità di rappresentare, allo stesso modo, sia le normali sequenze di eventi del processo, che le sequenze anomale dovute all'insorgere di un indesiderato evento. Infatti le connessioni causali che rendono conto, in termini di relazioni causa-effetto, sia delle sequenze normali che di quelle anomale, possono, in entrambi i casi, essere rappresentate come dipendenze di un GOAL (effetto) da altri SUBGOALS (cause). Si pensi, ad esempio, all'attributo SEQUENZA che contiene la lista dei SUBGOALS che normalmente vanno soddisfatti, o all'attributo CAUSE CHE PREVENGONO IL SUCCESSO che contiene i SUBGOALS imputabili del mancato successo.

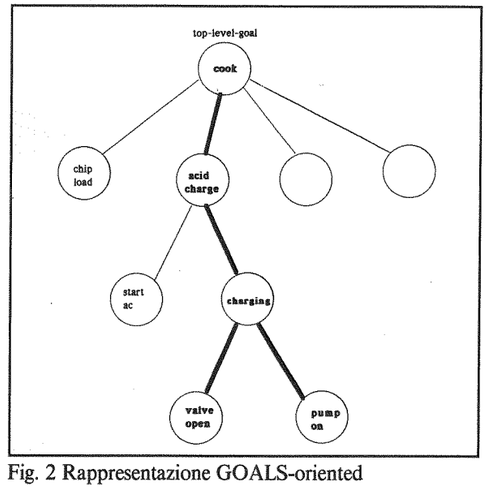


Fig. 2 Rappresentazione GOALS-oriented

DISCRIMINATORE DEL CONTESTO

Identifica il contesto in cui si trova il processo.

REQUISITI

Condizioni o GOALS che devono essere costantemente soddisfatte durante tutto il tempo necessario al soddisfacimento del GOAL.

SEQUENZA

Sequenza di SUBGOALS da soddisfare per il successo del GOAL.

CRITERI DI SUCCESSO

Condizioni che determinano il soddisfacimento.

CRITERI DI INSUCCESSO

Condizioni che confermano l'insuccesso.

CAUSE CHE PREVENGONO IL SUCCESSO

Possibili cause di insuccesso (GOALS).

PROBLEMA

Condizioni che generano un problema.

RIMEDI

Possibili rimedi al problema presentatosi.

Fig. 3 Attributi di una GOAL.

4.2 Rappresentazione dell'estensione temporale mediante storia e aspettazioni di ogni singolo dato

Ogni settore, quindi ogni dato, è anch'esso rappresentato come un oggetto, o frame, con i suoi attributi. L'insieme dei sensori è quindi mappato su un insieme di oggetti. Questo insieme è ripartito in classi. Ciascuna classe individua una ben precisa tipologia di oggetti e quindi di sensori. Al di là dei diversi attributi specifici di ogni tipo di sensore, gli attributi comuni a tutti sono

VALORE,
TIME-STAMP,
MAX-AGE,
HISTORY-LIST,
EXPECTATIONS

Essi contengono, in ordine, il valore del dato, il tempo in cui è stato rilevato il valore, il periodo di tempo dopo il quale il valore è considerato obsoleto, la storia degli ultimi "n" valori con il loro time-stamp, le aspettazioni sui futuri valori. Si veda l'esempio di un sensore di flusso in figura 4. Mentre i primi quattro attributi sono aggiornati dal sistema di acquisizione dati, le EXPECTATIONS sono aggiornate come side-effect del processo di esplorazione della rete di GOALS da parte dell'inference engine. In tal modo ogni dato possiede una storia passata ed una proiezione sulla vita futura. Si realizza così quella estensione temporale dei dati che è un presupposto fondamentale alla flessibilità del problem-solving.

VALORE

21.05 litri/min

TIME-STAMP

12-h, 31-min, 47-sec, 754-ms

MAX-AGE

0.5-sec

HISTORY-LIST

12:31:47:754, 21.05
12:31:47:305, 21.51
12:31:47:011, 21.89
12:31:46:684, 22.12
12:31:46:295, 22.73

EXPECTATIONS

Decrescente,
Decrescente non meno di 5 litri/min
Stabile alla chiusura

Fig. 4 Esempio di data frame

4.3 Gradi di flessibilità del problem-solving

Fatte le debite premesse sullo schema di rappresentazione della conoscenza GOALS-oriented (punto 4.1) e sul modo in cui viene garantita l'estensione temporale dei dati (punto 4.2), vediamo come il sistema affronta i problemi generali riguardanti la flessibilità richiesta al problem-solving discussi al punto 3. La tentazione di presentare l'argomento in modo esaustivo ci condurrebbe a diagrammi di stato-transizioni dei GOALS improponibili, preferiamo invece attenerci agli scopi di questo articolo, tentando di trasmettere un'idea della soluzione, rinviando che fosse interessato ad una più completa e rigorosa descrizione contenuta nel documento [11] disponibile presso l'autore.

È opportuno fare un distinguo tra quello che è la strategia generale del sistema e le modalità di utilizzo della conoscenza espressa negli attributi dei GOALS.

Per quanto riguarda la strategia generale, c'è da dire che il sistema alterna un ciclo di aggiornamento dati ad un ciclo inferenziale sulla rete dei GOALS. Il ciclo di aggiornamento dati, di cui non tratteremo in questo articolo è un meccanismo piuttosto complesso con criteri di "focusing" su di un subtest di dati, aggiornamenti per raggiunta obso-

lescenza, il tutto gestito dinamicamente e pilotato dai side-effects dell'inference engine. Per una descrizione esauriente di tali meccanismi si veda il documento [11].

Il ciclo inferenziale sulla rete di GOALS è invece meglio sintetizzabile in una pur così generale descrizione. In ogni ciclo inferenziale l'insieme dei GOALS della rete è ripartito in due sottoinsiemi: GOALS attivi e GOALS disattivi. Il compito dell'inference engine è quello dicercare di soddisfare, secondo le modalità indicate dagli attributi dei GOALS, tutti i GOALS attivi. In realtà, ad ogni ciclo, pochi GOALS saranno soddisfacibili, altri potranno anche essere disattivati, mentre i più continueranno a restare attivi e non soddisfacibili.

Veniamo ora ad esaminare invece come localmente, su ogni GOAL, viene utilizzata dall'inference engine la conoscenza espressa negli attributi. Si veda come riferimento la figura 3.

Il DISCRIMINATORE DEL CONTESTO contiene le condizioni che individuano il contesto entro il quale il GOAL ha un ruolo attivo. Con un eccesso di semplificazione potremmo dire che stabilisce quando il GOAL è attivo nel senso sopra descritto; ossia facente parte del sottoinsieme dei GOALS attivi. La funzione di questo attributo è di collocare contestualmente il GOAL prevenedone un utilizzo improprio.

I REQUISITI sono i SUBGOALS che devono essere preventivamente e costantemente soddisfatti prima e durante il tentativo di soddisfacimento del GOAL. Costituiscono una garanzia che si stia costantemente operando sullo stesso scenario. Se viene a mancare qualche requisito, non solo è inutile, ma potrebbe anche essere sbagliato perseguire il successo del GOAL.

La SEQUENZA è la lista dei SUBGOALS che vanno soddisfatti, nella relazione d'ordine indicata, per poter tentare di avere successo con questo GOAL.

CRITERI DI SUCCESSO e CRITERI DI INSUCCESSO sono le condizioni che dichiarano rispettivamente soddisfatto o fallito il GOAL. È da notare che il GOAL può sopravvivere attivo pur non essendo ne soddisfacibile, ne dichiarabile fallito. In tal caso, se sono verificate le condizioni contenute nell'attributo PROBLEMA, il non soddisfacimento del GOAL provoca la generazione di un problema.

CAUSE CHE PREVENGONO IL SUCCESSO contiene i SUBGOALS imputati di causare l'insuccesso. applicando i RIMEDI di tali SUBGOALS, ossia fornendo all'operatore i suggerimenti sulle azioni da intraprendere, sarà in generale possibile risolvere il problema generato. Se il problema dovese persistere, si aumenta il livello di indagine. Quasi sempre questo implica il prendere in considerazione RIMEDI che coinvolgono l'operatore in una sessione interattiva di problem solving. È naturalmente desiderabile rendere il sistema il più autonomo limitando l'intervento dell' operatore a quei casi in cui se non se ne può fare a meno. Per questo motivo l'attività di problem solving è organizzata in diversi livelli di profondità di indagine. Ai livelli più profondi si trovano le richieste di informazioni e di collaborazione verso l'operatore. Tali livelli sono attivati solo quando i più superficiali falliscono.

Dalla descrizione fatta dovrebbe risultare comprensibile come, attraverso la possibilità di mantenere in vita GOALS non soddisfacibili che però non necessariamente generano problemi, condizioni contestuali esplicite che evitano l'uso improprio dei GOALS, prerequisiti da soddisfare che evitano al problem-solving di continuare con gli occhi bendati una volta individuata una strada da seguire, sia possibile dare al sistema quei gradi di libertà necessari ad operare secondo il tipo di flessibilità auspicata. Non c'è molto da dire sull'interfaccia utente, se non che è del tipo a finestre. Più interessante è il trattamento delle richieste asincrone che il sistema invia all'operatore, ma ancora una volta rimandiamo al documento [11] per una dettagliata descrizione che esula dai contenuti di questa nota.

5. CONCLUSIONI

La conclusione del progetto con l'installazione di un sistema esperto presso una realtà industriale e la sua accettazione da parte degli utenti può senza dubbio considerarsi un punto positivo. È auspicabile che i problemi sollevati, e non completamente risolti, nello sviluppo di una architettura adatta a problematiche real time, serviranno come argomento di ulteriore indagine per la creazione di soluzioni sempre più adeguate.

6. BIBLIOGRAFIA

[1] J.F. Allen, TOWARDS A GENERAL MODEL OF ACTION AND TIME, Artificial Intelligence 23, 2, July 1984

[2] J.F. Allen, P.J. Hayes, A COMMON-SENSE THEORY OF TIME, IJCAI 1985, Proceedings, pp. 528-531

[3] R. Turner, LOGICS FOR ARTIFICIAL INTELLIGENCE, (Temporal logic in AI) John Wiley and Sons, pp. 77-100, 1984

[4] J. Malik, T.O. Bindford, REASONING IN TIME AND SPACE, IJCAI 1983, Proceedings, pp. 343-345

[5] J.W. Long, T.A. Russ, A CONTROL STRUCTURE FOR TIME DEPENDENT REASONING, IJCAI 1983, Proceedings, pp. 230-232

[6] S.M. Fox, S. Lowenfeld, P. Kleinosky, TECHNIQUES FOR SENSOR-BASED DIAGNOSTICS, IJCAI 1983, Proceedings, pp. 158-163

[7] R. Sauers, R. Walsh, ON THE REQUIREMENTS OF FUTURE EXPERT SYSTEMS, IJCAI 1983, Proceedings, pp. 110-115

[8] R. Moore, L. Hawkinson, C. Knickerbocker, L. Churchman, A REAL-TIME EXPERT SYSTEM FOR PROCESS CONTROL, IEEE Conference on AI application 1984, Proceedings, pp. 569-576

[9] M. Gallanti, G. Guida, L. Spampinato, A. Stefanini, REPRESENTING PROCEDURAL KNOWLEDGE IN EXPERT SYSTEMS: AN APPLICATION TO PROCESS CONTROL, IJCAI 1985, Proceedings, pp. 345-352

[10] J.H. Greismer, S.J. Hong, M. Karnaugh, J.K. Kastner, M.I. Schor, R.L. Ennis, D.A. Klein, K.R. Milliken, H.M. VanWoerkom, YES/MVS: A CONTINUOUS REAL TIME EXPERT SYSTEM, AAAI 1984, Proceedings, pp. 130-136

[11] J. Allard, E. Bertolotti, W. Kaemmerer, J. Nomura, COOKER DESIGN DOCUMENT, disponibile in HI-SI-Pregnana, presso l'autore, 1986

RESCU: UN SISTEMA ESPERTO IN REAL TIME

Alessandro Mazzetti

Systems Designer Spa - Expert Systems Application Centre

Milano

RIASSUNTO

In questo articolo viene presentato RESCU, uno Shell di Sistema Esperto per applicazioni in tempo reale, nato da una collaborazione interaziendale ed universitaria sotto gli auspici del programma di ricerca ALVEY. Dopo una breve descrizione dell'ambito applicativo in cui è stato sviluppato il prototipo, vengono analizzate le motivazioni della scelta di un Sistema Esperto piuttosto che un sitema tradizionale e vengono descritti gli strumenti utilizzati. Si passa poi all'analisi dell'architettura e delle caratteristiche del sistema, con particolare attenzione verso i meccanismi inferenziali e della meta-conoscenza. Segue un esempio di funzionamento in ambito di controllo di qualità di prodotti chimici. L'esperto termina con un esame dei risultati conseguiti e delle prospettive future.

ABSTRACT

This paper describes RESCU, an Expert System Shell for real time application, developed by a collaborative club of companies and universities, under the auspices of the ALVEY research programme. After a brief description of the applicative environment, the motivations of the choice of an Expert System rather a conventional system are analysed and the development tools are described. The system architecture and features are then shown, with particular attention towards the inferential and meta-knowledge mechanisms. An example follows, in the field chemical quality control. The paper ends with an evaluation of the results and of some future developments.

1. INTRODUZIONE

RESCU è un acronimo per "Real time Expert System Club of Users". La sigla rappresenta quindi un club di utenti, per la precisione 21 tra le maggiori aziende industriali inglesi e tre Università, che hanno finanziato un progetto per lo sviluppo di un Sistema Esperto per il controllo di processo in tempo reale.

La motivazione di una simile impresa è il fatto che gli impianti industriali stanno diventando sempre più grandi e complessi, di conseguenza anche i sistemi computerizzati di controllo stanno assumendo proporzioni difficilmente gestibili. Si sta andando, dunque, verso una situazione in cui l'efficienza di un impianto sia tutta nelle mani di pochi esperti capaci di manovrare le complesse apparecchiature di controllo. Aggiungendo a ciò la tendenza a ridurre sempre di più le presenze umane in fabbrica, si ottengono valide motivazioni per automatizzare ulteriormente i sistemi di controllo. Il problema è che un'autonomia di questo livello richiede una notevole dose di "esperienza" e "conoscenza" dell'impianto.

È dunque sentita la necessità di un Sistema Esperto in grado di sostituire (o aiutare) l'operatore d'impianto, vale a dire in grado di prendere autonomamente alcune decisioni sul "come" far funzionare l'impianto. Uno dei punti più difficili per tale tipo di sistemi è la necessità di dover lavorare in tempo reale, raccogliendo periodicamente informazioni dai sensori di impianto e facendone un uso intelligente e tempestivo.

In passato nessun gruppo industriale aveva osato affrontare in proprio questo tema, perchè la tecnologia non era ancora matura e si prevedevano eccessivi investimenti in ricerca. Il Governo di Londra decise invece di incoraggiare lo sviluppo delle nuove tecnologie, avviando il progetto ALVEY (equivalente a livello Britannico del progetto ESPRIT europeo). Nacque dunque RESCU verso la metà del 1984 su proposta della società Systems Designers, le

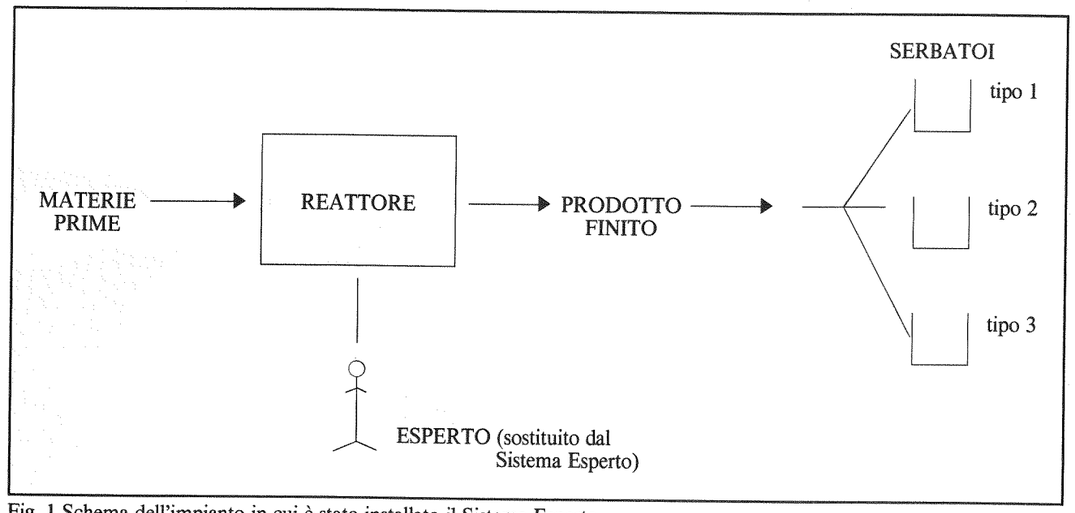


Fig. 1 Schema dell'impianto in cui è stato installato il Sistema Esperto

cui esperienze applicative erano: un Sistema Esperto per la concessione di credito bancario, sviluppato in SAGE per una banca Belga [7]; un Sistema Esperto per la stima dei costi di impianti di telecomunicazione [8]; un Sistema Esperto per il disinnescamento di bombe, per il Ministero della Difesa Britannico.

Oggi quello di RESCU è il solo Sistema Esperto di questo tipo funzionante in Inghilterra. L'intento del progetto è non solo quello di capire e valutare l'applicabilità dei Sistemi Esperti al controllo in Real Time dei processi industriali, ma anche quello di esplicitare le problematiche manageriali ed organizzative presenti in una collaborazione interaziendale.

2. L'AMBITO APPLICATIVO

La scelta dell'applicazione è caduta sullo sviluppo di un Sistema Esperto per il controllo di qualità di un impianto chimico del Nord-est della Gran Bretagna. Si tratta di una fabbrica di detergenti industriali che produce diversi tipi di detergente a partire dalle medesime materie prime e variando solo alcuni parametri nella reazione (vedi figura 1). A secondo del tipo di prodotto ottenuto, questo verrà immagazzinato in diversi serbatoi, uno per ogni tipo.

Il ciclo di lavorazione avviene a lotti e ogni serbatoio può contenere parecchi lotti. Supponiamo che venga ordinata una certa quantità di detergente, ad esempio del tipo 2: sarà necessario effettuare più volte la reazione e riempire il serbatoio 2 man mano con i vari lotti. Come già accennato, per ottenere un prodotto di un certo tipo, è necessario agire su alcuni parametri di reazione: ad esempio, la quantità degli ingredienti, la loro densità, la temperatura, i tempi di reazione, etc.

Il problema è quindi rappresentato dalla regolazione di tali parametri da parte degli operatori d'impianto, esso è particolarmente difficile e richiede una notevole esperienza. In altre parole, basta una minima fluttuazione di un parametro, diciamo la densità, per ottenere un prodotto di qualità totalmente differente, ad esempio del tipo 3 anziché 2. A questo si aggiunge che le materie prime non sono sempre della stessa qualità, quindi nella regolazione del reattore bisogna tenerne conto. Inoltre l'effetto della regolazione non è immediato, ma può risentire di una certa inerzia: ad esempio per diminuire la densità di un reagente, non basterà chiudere la valvola al momento desiderato, ma bisognerà chiuderla un po' prima, perchè la reazione ha un po' di inerzia e continuerà per un po' di tempo anche con la valvola chiusa. Se poi l'operatore non fosse riuscito a regolare la reazione in modo perfetto ed avrà ottenuto un prodotto scadente, potrà sempre compensare l'errore durante il lotto successivo, cercando di ottenere un prodotto scadente in senso opposto che, mescolato con quello precedente nel serbatoio, costituirà globalmente il prodotto desiderato.

3. PERCHÈ UN SISTEMA ESPERTO

In teoria, la regolazione dei parametri di reazione potrebbe essere descritta da un modello matematico, che tenesse conto di tutte le variabili in gioco e fornisse i valori esatti di regolazione delle valvole e delle apparecchiature; l'operatore d'impianto opera invece spesso sulla base di sensazioni ed effettua delle regolazioni "a naso" dettate dalla sua pluriennale esperienza. Addirittura, può capitare che l'operatore si accorga che un sensore o uno strumento di misura sia errato, semplicemente notando certe discrepanze fra i valori misurati e certe evidenze nell'andamento dell'impianto. Ma potrebbe anche capitare che certi comportamenti anomali dell'impianto si manifestino solo in alcune situazioni particolari, in cui non fosse necessaria una ottimizzazione perfetta, di conseguenza non varrebbe

la pena di intraprendere alcuna azione.

In altre parole non è ottimale lo studio di un modello matematico dell'esatto comportamento dell'impianto, sia perchè sarebbe troppo complesso, sia perchè l'esperienza a tutt'oggi disponibile degli operatori di impianto può essere più significativa nella pratica. Ecco dunque il motivo per cui si è preferito affrontare questo problema con tecniche di Intelligenza Artificiale anziché con tecniche tradizionali: i Sistemi Esperti, in effetti sono particolarmente adatti per immagazzinare esperienze non derivate da modelli matematici, ma provenienti dalla pratica di tutti i giorni [1].

La grossa difficoltà rispetto alla maggior parte dei Sistemi Esperti oggi funzionanti si trova, però, nel fatto che si tratta di un'applicazione in real-time, cioè in cui le variabili in gioco si modificano nel tempo. Il Sistema Esperto deve quindi essere in grado di accorgersi di eventuali mutamenti occorsi nell'arco delle sue inferenze ed essere in grado di guidare di conseguenza le inferenze stesse. Nel caso, ad esempio, in cui un certo parametro fosse in rapida crescita, il sistema deve poter prendere una decisione entro un ben determinato tempo, quindi è necessaria una analisi preliminare per evitare di perdere tempo in indagini inutili: si tratta dunque di una sorta di ragionamento per decidere quale ragionamento effettuare, detto anche meta-ragionamento.

La Systems Designers aveva già sviluppato il pacchetto MXA, che avrebbe potuto essere utilizzato: si tratta di uno Shell di Sistema Esperto in real-time particolarmente evoluto, sviluppato in ambito militare in interpretazione delle immagini provenienti dagli aerei. Pur essendo strutturato a "blackboard" e progettato per operazioni in temporeale [5], MXA è però rivolto ad applicazioni dove è dominante una strategia del tipo "ipotesi e test" a causa di uno spazio infinito di soluzioni legate a insiemi di istanze simili, mentre nel nostro caso è presente solo un'istanza di ogni elemento dell'impianto.

Si è così concluso che fosse più semplice partire da zero per quello che concerneva la macchina inferenziale, cercando però di impiegare linguaggi e strumenti adatti per una applicazione complessa di intelligenza artificiale. La scelta cadde su PROLOG [3], il noto ambiente di programmazione concepito nelle università di Edimburgo e del Sussex e commercializzato dalla Systems Designers. PROLOG permetteva di legare insieme parecchi vantaggi tra i quali la portabilità su molte macchine (dalla famiglia dei VAX alle workstation SUN e APOLLO, agli HP e in genere alle macchine basate sul 68000) e la potenza di linguaggi quali il PROLOG [10], il LISP ed il POP11 in uno stesso ambiente. In particolare il linguaggio POP11 [4] si dimostrava particolarmente adatto a questo tipo di applicazione, in quanto congiunge l'elasticità dei linguaggi funzionali/dichiarativi con la ricchezza di strutture dei linguaggi tipo ALGOL e PASCAL.

Purtroppo all'epoca non era ancora disponibile il MicroVAX/2, con tutti i vantaggi che questa macchina avrebbe implicato, per cui la scelta è caduta su un VAX 11/730

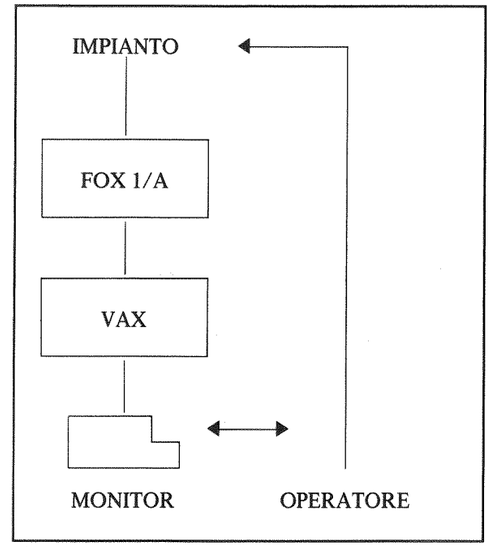


Fig. 2 Connessione impianto-computers

sotto VMS, con un monitor a colori come interfaccia verso gli operatori dell'impianto. Il VAX ha anche il compito di raccogliere i dati di interesse dialogando con un calcolatore di processo FOX 1/A della FOXBOROUGH, già presente nell'impianto. Il sistema è per ora soltanto consultativo [12], nel senso che il suo output non è mai diretto sull'impianto, ma è solamente rivolto all'operatore che dovrà aiutare le operazioni indicate dal sistema sul monitor (figura 2). L'input è invece misto: alcune informazioni vengono reperite direttamente dai sensori nell'impianto, mentre altre vengono chieste all'utente attraverso domande poste sul monitor. Queste scelte, sia per l'hardware che per il software, avevano anche importanza in quanto permettevano ai membri del club di visionare i risultati raggiunti ad un costo relativamente basso in termini di tempo.

4. ARCHITETTURA E CARATTERISTICHE DEL SISTEMA

Come la maggior parte dei Sistemi Esperti oggi più diffusi, il sistema è composto da una base di conoscenza, un motore inferenziale e una interfaccia utente. A differenza dei Sistemi Esperti tradizionali, vi sono canali di comunicazione per i dati trasmessi dal VAX al calcolatore di processo e viceversa per lo scambio di informazioni relative al processo in corso nell'impianto.

È stato studiato e definito un apposito linguaggio per la rappresentazione della conoscenza, denominato semplicemente KRL-Knowledge Representation Language. Si tratta di un linguaggio basato su regole piuttosto che su frame [6], dato che la Systems Designers aveva già una notevole esperienza in linguaggi di tale genere con SAGE [11] ed ENVISAGE [2].

In una base di conoscenza espressa in KRL le regole sono suddivise in due moduli: "attivi" e "passivi", denominati "sorgenti di conoscenza". Le sorgenti attive raccolgono solo la parte procedurale del sistema, cioè le "istruzioni" da eseguire senza l'implicazione di particolari meccanismi inferenziali. Corrispondentemente, le sorgenti passive raccolgono solo la parte dichiarativa del problema, vale a dire le regole vere e proprie utilizzate dal motore inferenziale per la valutazione delle sue ipotesi. La forma di tali regole è l'usuale "IF condizione THEN azione", in cui nella parte di condizione le variabili possono essere collegate da operatori booleani per formare espressioni logiche. Esiste un meccanismo per accettare regole nidificate, del tipo

IF condizione-1 THEN

IF condizione-2 THEN

azione

utile per evitare la valutazione ripetitiva di precondizioni comuni a un numero di diverse regole. Questa situazione può occorrere ad esempio nel caso di regole che esprimano fasi correlate di un lotto di produzione (o "batch").

Le regole possono avere associate probabilità che influenzano il comportamento delle variabili (dette attributi); le variabili stesse possono avere delle probabilità bayesiane che vengono calcolate nella parte di azione delle regole. Ogni variabile può avere o non avere un valore associato. Se non ce l'ha, la variabile prende il nome di "indefinita" e quando le verrà fatto qualche riferimento, verrà "acceso" il motore inferenziale allo scopo di dedurre tale valore mancante. Chiaramente non possono essere dedotti valori di variabili che rappresentano oggetti misurati.

Per poter far fronte alle esigenze real-time dell'applicazione, si è rivelato non sufficiente associare alle variabili un unico valore, ma è stato necessario gestire una sequenza di valori assegnati nei diversi momenti in cui la variabile viene valutata. In questo modo, ogni variabile si "porta dietro" tutta la storia della sua evoluzione temporale, nei cosidetti "slots" associati al tempo di campionamento ("scan"). Tale tipo di variabili viene utilizzato nelle regole attraverso costrutti che suonano così:

IF X al momento Y valenza Z THEN

probabilità__di__K = P

oppure

IF incremento__di__X > Y% in questi ultimi Z secondi

THEN chiudere la valvola K

Il linguaggio KRL ha come caratteristica quella di essere aperto verso linguaggi esterni per cui è possibile utilizzare al suo interno routines scritte in POP11 o, di conseguenza, in qualsiasi altro linguaggio; questo ha come vantaggio la possibilità di spostare lunghe computazioni su linguaggi più adatti o far manipolare strutture dati complesse da sistemi più idonei. Questo non significa però che vi sia carenza di strutture matematiche; anzi, una delle più interessanti caratteristiche del KRL è la "reversibilità" delle equazioni. In altre parole, se una regola contiene una qualsiasi espressione matematica, ad esempio:

A = (B * C)—D

il compilatore genererà in modo automatico tutte le possibili equazioni invertite:

B = (A + D) / C

C = (A + D) / B

D = (B * C)—A

allo scopo di poter utilizzare l'informazione contenuta in quella equazione in tutti i possibili modi. Si noti che l'inversione delle equazioni viene fatta a livello simbolico e non numerico.

5. MECCANISMI DI INFERENZA E DI META-CONOSCENZA

Per quello che concerne il motore inferenziale, vi sono quattro meccanismi di inferenza:

— backward chaining singolo (default)

— forward chaining singolo

— backward chaining multiplo

— forward chaining multiplo

I meccanismi di inferenza singola svolgono funzioni analoghe a quelle dei più comuni Sistemi Esperti. Il backward chaining singolo, che è quello di default, viene invocato automaticamente quando è richiesto il valore di una variabile "indefinita"; gli altri tre meccanismi di inferenza, invece, devono essere invocati esplicitamente attraverso appositi comandi del KRL. Il forward chaining singolo viene usato per ottenere le conseguenze di un insieme di valori, applicando le regole al contrario rispetto al bakward chaining singolo.

I meccanismi di inferenza multipla servono essenzialmente per fare delle verifiche di consistenza sulle valutazioni ottenute: si tenga presente che, trattandosi di un Sistema Esperto in real-time, potrebbero verificarsi delle inconsistenze dovute alla naturale evoluzione temporale dell'impianto. Il backward chaining multiplo serve dunque per controllare la consistenza delle ipotesi nella deduzione di un dato valore, mentre il forward chaining multiplo controlla la consistenza di un insieme di valori date le ipotesi.

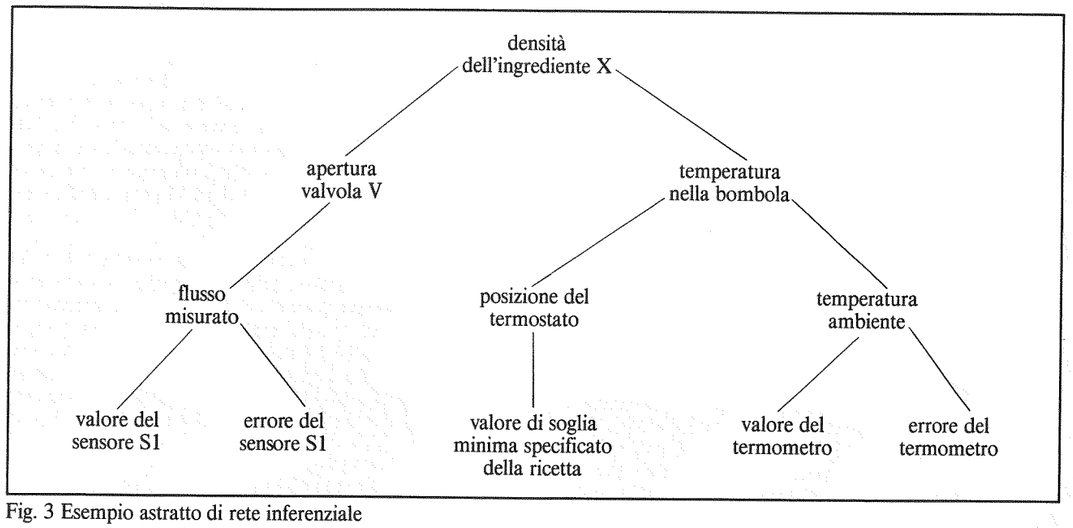


Fig. 3 Esempio astratto di rete inferenziale

Vi sono, inoltre, alcuni meccanismi di meta-conoscenza per il controllo dell'inferenza, fra cui:

— avviare e fermare la generazione di ipotesi;

— definire gli obiettivi per il backward chaining;

— stabilire la priorità delle ipotesi.

Esiste, infine, un meccanismo di spiegazione molto utile in fase di utilizzo (nonché di debugging) soprattutto per permettere all'utente di decidere se il suggerimento fornito è basato su dati fondati oppure completamente fuori luogo; potendo di conseguenza operare con confidenza sullo strumento fornito.

Per capire meglio il ruolo di questi meccanismi di inferenza e di meta-conoscenza, facciamo un esempio di esecuzione. Consideriamo una rete inferenziale astratta e semplice da capire, come quella di figura 3. È rappresentato il seguente concetto: la densità dell'ingrediente X dipende dall'apertura della valvola V e dalla temperatura della bombola; la valvola è regolata da un meccanismo automatico che opera in base al flusso misurato nel tubo, mentre la temperatura della bombola è mantenuta costante da un meccanismo dotato di termostato.

Quando il Sistema Esperto è in opera, si trova in continuo loop in cui ogni cinque minuti fa un campionamento (scan) delle variabili di impianto, rilevate da appositi sensori (la scelta di cinque minuti è dovuta ai tempi intrinsecamente lenti della reazione chimica). Si supponga che il sensore che misura la densità dell'ingrediente X dia un certo valore e che sulla base di alcune "costanti di confronto" il sistema si "accorga" che tale valore è anomalo rispetto a quello che dovrebbe avere in situazione di normalità. In altre parole, il sistema ha scoperto una "discrepanza" fra quella che è la situazione reale e quella che dovrebbe essere.

Entra dunque in gioco la rete inferenziale di figura 3, sulla base della quale, il sistema si pone come obiettivo di andare a vedere la causa della discrepanza. Per prima cosa il sistema cerca quali sono i fattori che influenzano il valore anomalo "scendendo" nell'albero fino a raggiungere le "foglie" (valore ed errore del sensore S1; valore di soglia della ricetta; valore ed errore del termometro); a questo punto viene "acceso" il motore inferenziale forward chaining singolo per dedurre quale avrebbe dovuto essere il valore corretto. Se viene confermato che tale valore è differente da quello misurato, allora si assume che la causa della discrepanza sta in qualche fattore intermedio, cioè in qualche nodo intermedio fra la radice e le foglie dell'albero (nel nostro caso, ad esempio, può essere la valvola che non funziona bene o magari la bombola che perde). Vengono quindi usate le regole di meta-conoscenza per generare un insieme di ipotesi candidate (ad esempio, "quello strumento potrebbe essere in errore") e viene "acceso il backward chaining multiplo per percorrere l'albero verso il basso e individuare il nodo responsabile della discrepanza. A questo punto entrano in gioco eventuali altre regole di meta-conoscenza per prendere una decisione sul come proseguire: si potrebbe dare all'utente una semplice segnalazione, oppure indicare un'azione da intraprendere (ad esempio "chiudi a mano la valvola V"), oppure accantonare il problema per risolvere problemi più urgenti sorti nel frattempo.

Questo semplice esempio ha messo in luce aspetti più importanti del KRL e dei suoi meccanismi inferenziali. La base di conoscenza reale del sistema è composta essenzialmente da regole di tale tipo ed è strutturata in modo da rispecchiare un'architettura naturale per il tipo di applicazione. È infatti composta da diverse parti fra cui:

— un modello dell'impianto,

— una traccia del monitoraggio del processo,

— alcune regole per un migliore controllo sulla qualità del prodotto,

— le ricette che descrivono i composti chimici da utilizzare per ogni tipo di prodotto finale.

6. RISULTATI E PROSPETTIVE FUTURE

Oggi il sistema è in funzione presso lo stabilimento, dove già sta apportando benefici in termini di qualità della produzione. L'utente finale del sistema è l'operatore di impianto che riceve sul monitor la segnalazione delle decisioni prese dal sistema e attua le operazioni. Il sistema non è considerato definitivo: sono ancora in fase di studio alcuni perfezionamenti e alcune aggiunte. Una caratteristica che si vorrebbe inserire nel prototipo successivo è quella dell'"inheritance" tipica in ambiente di programmazione orientata agli oggetti ("object oriented programming" [9]) e quindi di strutture di tipo "frame";questo permetterebbe la formulazione, ad esempio, di una regola relativa agli errori generica per tutti gli strumenti invece di avere per ogni strumento una specifica regola (per esempio, regole del tipo: "valore di strumento = valore misurato + errore di strumento").

La fase di acquisizione della conoscenza non è stata particolarmente impegnativa, a differenza degli usuali progetti nel campo dei Sistemi Esperti, in cui il maggior sforzo è sempre dedicato all'elicitazione della conoscenza. Questo può essere dovuto al fatto che le altre fasi sono state lunghe ed impegnative (definizione del KRL e l'implementazione del relativo compilatore), quindi hanno messo in secondo piano quella della acquisizione della conoscenza.

Il KRL è stato sviluppato implementando un compilatore, un linker e il run-time system. Questo perchè doveva essere uno strumento che permettesse a tutti i membri del club di sviluppare applicazioni nel campo dei Sistemi Esperti in tempo reale, sfruttando le esperienze di questa fase di prototipizzazione. Per questo motivo ogni membro del club è stato costantemente mantenuto informato sull'andamento del progetto attraverso la distribuzione di "working papers" che fissavano la varie fasi del progetto non solo da punti di vista tecnici (come erano fatte le cose) ma anche sotto il profilo di "scheduling" dei passi di lavoro (come e perchè erano interconnesse le varie sottofasi). In questo modo si è colto l'obiettivo di mostrare il processo di disegno ed implementazione del sistema ed è stato grazie a questa metodologia di lavoro che si sono potuti affinare gli obiettivi e i mezzi per conseguirli.

I risultati del progetto RESCU sono molteplici e piuttosto difficili da valutare: in prima battuta si può considerare come risultato il particolare sistema oggi già in funzione presso lo stabilimento. I benefici che sta apportando sono soprattutto in termini di segnalazione tempestiva e rigorosa delle anomalie nascoste che possono verificarsi durante la reazione chimica: oggi il prodotto finale del reattore è di qualità molto migliore di quello ottenuto in precedenza. Ma in un contesto più ampio, il risultato più interessante è stato il fatto di aver realizzato uno Shell innovativo per Sistemi Esperti Real-time in grado effettivamente di far fronte ad applicazioni reali, e questo grazie al fatto che è stato studiato e realizzato direttamente in un ambito applicativo. Un altro risultato è la metodologia di sviluppo adottata in questo progetto, che ha dimostrato la concretezza di un club collaborativo in cui impegni ed esiti sono condivisi.

Sulla scia di questi risultati, i membri del club RESCU prevedono un vasto uso dello Shell realizzato, soprattutto per applicazione di controllo o gestione di impianti, in quei settori dove è richiesto un notevole grado di esperienza nelle scelte di funzionamento. Si prevedono anche applicazioni nel campo dell'analisi degli allarmi e di situazioni di errore o malfunzionamento, per automatizzare le azioni di ripristino in modo intelligente ed ottimale.

7. RINGRAZIAMENTI

Si ringrazia l'Alvey Directorate sia per aver aiutato ed incoraggiati lo sviluppo del progetto, sia per averne autorizzata la citazione. Si ringraziano particolarmente l'Ing. Carlo Tarantola per i contributi tecnici e Mr. Anthony Wilson per i suoi preziosi consigli. Un ringraziamento anche a Ray Shaw, David Shorter e A. Wheldon per i loro contributi.

8. BIBLIOGRAFIA

[1] S. Bandini, I SISTEMI ESPERTI, Note di Software n. 23/24, 1984

[2] A. Mazzetti, APPLICAZIONI DEI SISTEMI ESPERTI, Muzzio Ed., 1987

[3] S. Hardy, A NEW SOFTWARE ENVIRONMENT FOR LIST PROCESSING AND LOGIC PROGRAMMING, Artificial Intelligence: Tools Techniques Applications, Harper and Row, 1984

[4] R. Barret, A. Ramsay, A. Sloman, POP-11, A PRACTICAL LANGUAGE FOR AI, Ellis Horwood and Wiley, 1985

[5] R.A. Stammers, THE MXA SHELL, Research and Developments in Expert Systems, Cambridge University Press, 1984

[6] M. Minsky, A FRAMEWORK FOR REPRESENTING KNOWLEDGE, The Psycology of Computer Vision, (P. Winston Ed.), McGraw Hill, 1975

[7] A. Mazzetti, APPLICAZIONI REALI IN AMBITO BANCARIO, Zerouno, Novembre 1986

[8] A. Mazzetti, UN SISTEMA ESPERTO DI SUCCESSO TUTTO ITALIANO, Informatica Oggi, Gennaio 1987

[9] M. Stefik, D. Bodrow, OBJECT-ORIENTED PROGRAMMING: THEMES AND VARIATIONS, The AI Magazine, Vol. 6 No 4, 1986

[10] C. Mellish, S. Hardy, INTEGRATING PROLOG INTO THE PROLOG ENVIRONMENT, Proceedings of the 8th IJCAI Conference, Karlsruhe, 1983

[11] M. Borghesi, M. Gallanti, A. Stefanini, AMBIENTI SVILUPPO PER SISTEMI ESPERTI, Note di Software n. 23/24, 1984

[12] J.F. Glimore, C. Howard, EXPERT SYSTEMS TOOL EVALUATION, 6th International Workshop on Expert Systems and Theyr Applications, Avignon, 1986

[13] F. Hays-Roth, D.A. Waterman, D.B. Lenat, BUILDING EXPERT SYSTEMS, Addison Wesley, Reading, MA, 1983

[14] D. Michie, EXPERT SYSTEMS IN THE MICROELECTRONIC AGE, Edimburgh University Press, 1979

[15] D. Michie, INTRODUCTORY READINGS IN EXPERT SYSTEMS, Gordon and Breach Science Publisher, 1982

[16] M. Stefik et al., THE ORGANIZATION OF EXPERT SYSTEMS: A PERSPECTIVE TUTORIAL, Xerox, Palo Alto, CA, 1982

[17] A. Mazzetti, COSTRUIRE UN SISTEMA ESPERTO, Muzzio Ed., 1986

RECENSIONI

LOGIC PROGRAMMING, FUNCTIONS, RELATIONS AND EQUATIONS, ed. D. DeGroot e G. Lindstrom, Prentice Hall, 1986

Le raccolte di articoli di ricerca spesso non risultano adeguate ai compiti per cui sono state compilate: sovente non sono tempestive oppure raccolgono articoli di argomento troppo disparato e quindi mancano di un certo legame logico tra le parti. Questo volume, edito da due dei maggiori esperti del settore, non presenta questi difetti: esce infatti in un momento di grande fermento nel movimento della programmazione logica e raccoglie articoli che presentano diversi tentativi per risolvere uno dei problemi principali del Prolog: quello del trattamento dell'uguaglianza e il suo legame con l'amalgazione tra linguaggi logici e funzionali.

Lungi dall'essere limitato al Prolog, capostipite dei linguaggi logici, il "movimento„ della programmazione logica è infatti molto critico nei confronti di quest'ultimo linguaggio (considerato poco più di un assembler evoluto) ed è alla continua ricerca di miglioramenti e di nuovi approcci che ne eliminino i vari punti deboli che lo allontanano dagli obiettivi iniziali, primi fra tutti il problema del controllo (backtracking del Prolog) e il problema dell'uguaglianza.

Entrambi questi problemi sono cruciali per il futuro della programmazione logica, però, mentre il problema del controllo è maggiormente legato ad aspetti algoritmici e a miglioramenti tecnici dell'hardware (macchine parallele ecc.), il problema dell'uguaglianza è più profondo in quanto il Prolog non consente di dichiarare e trattare efficientemente l'uguaglianza semantica di due termini. Questo problema, inizialmente legato al filone di ricerca della dimostrazione automatica di teoremi, in cui sono state sviluppate molte tecniche, è strettamente legato al problema del controllo a causa dell'esplosione combinatoria che provoca nello spazio di ricerca delle possibili soluzioni. La logica equazionale è in ogni caso un aspetto importante della programmazione logica, e a questo aspetto sono stati dedicati molti sforzi da parte di numerosi ricercatori.

Un aspetto del problema precedente è quello della combinazione dei linguaggi logici e di quelli funzionali, tema principale di questo volume, che ha una notevole importanza pragmatica nella pratica della programmazione, in quanto, spesso, l'approccio funzionale risulta il più adatto per affrontare un certo problema. La programmazione logica ha in comune con la programmazione funzionale molti punti importanti, inclusa la natura applicativa e l'uso della ricorsione. Tuttavia vi sono anche notevoli differenze, principalmente riguardanti il trattamento della variabili che porta alla direzionalità in input-output dei linguaggi funzionali e che è assente nei linguaggi logici, l'esecuzione non deterministica del Prolog e i combinatori di ordine superiore, tipici dei moderni linguaggi funzionali, che danno a questi una maggiore espressività.

È perciò auspicabile, secondo molti ricercatori, e in questa raccolta si hanno molti argomenti a supporto di questa tesi, la combinazione di questi due paradigmi computazionali. Di fatto, sono già stati effettuati numerosi tentativi in questa direzione, con combinazioni più o meno complete dei due paradigmi e questo volume passa in rassegna alcuni dei risultati e delle proposte più interessanti: immissione di costrutti Lisp-like in Prolog, arricchimento di linguaggi funzionali con l'uso di variabili logiche, trattamento delle equazioni medianti il narrowing; ma non considera le combinazioni asimmetriche nelle quali si ha semplicemente il collegamento di due linguaggi come il Lisp e il Prolog che mantengono la loro natura, l'uno di linguaggio funzionale, l'altro di linguaggio logico.

Il volume è diviso in sei parti: una introduzione che presenta le problematiche di base tra linguaggi logici e funzionali e che dà le motivazioni di base che sottostanno a queste ricerche e una interessante classificazione delle varie strade possibili; una seconda parte riguardante l'aggiunta delle variabili logiche e dell'unificazione ai linguaggi funzionali, in cui vengono presentati i linguaggi HASL di Abramson, QUTE di Sato e Sakurai e FUNLOG di Subrahmanyam e You; vengono poi le combinazioni simmetriche, in cui vengono tentati degli approcci egualitari con LEAF di Barbuti, Bellia, Levi e Martelli e APPLOG di Cohen; segue un parte in cui viene presentato l'approccio della logica equazionale, trattata, dal punto di vista operazionale, mediante l'algoritmo di narrowing o tecniche analoghe, in cui vengono presentati Prolog-with-equality di Kornfeld, EQLOG di Goguen e Meseguer, TABLOG di Malachi Manna e Waldinger; altro approccio è quello dell'unificazione aumentata, che vede nell'unificazione il punto focale della programmazione logica e della sua integrazione con la programmazione funzionale; in questa parte viene presentato UNICORN di Bandes e UNIFORM di Kahn' da ultimo vengono affrontati i fondamenti semantici delle amalgame logico funzionali in cui vengono presentati gli aspetti teorici del problema da parte di Jaffar Lassez e Maher e con FRESH di Smolka.

Questa rassegna di lavori per l'attualità e l'indiscutibile interesse merita la sua collocazione sul tavolo di lavoro di coloro che si occupano a vario titolo di linguaggi logici e funzionali e di chi desidera approfondire il problema dell'amalgama tra questi due paradigmi computazionali, studiando direttamente articoli di ricerca originali, spesso di difficile reperimento. Il fatto che non vengono considerate le proposte di amalgame asimmetriche, come ad esempio LOGLISP e H non riduce l'interesse di questo volume che presenta gli approcci più recenti in questo settore e che probabilmente avranno un maggior sviluppo in futuro. Stabilire quale di queste varie proposte sia la più interessante non è compito da prendere a cuor leggero: le varie proposte hanno un proprio contenuto di originalità e quindi in futuro si vedrà quale di queste avrà il maggior successo. Il tempo e probabilmente altre integrazioni ancor più spinte e egalitarie, ci daranno quel linguaggio logico-funzionale che tutti noi speriamo veda presto la luce. Nell'attesa questo volume è uno sprone verso la ricerca di soluzioni migliori di queste. Si deve anche notare che di alcuni dei linguaggi presentati vengono riportati i sorgenti degli interpreti (in Prolog dimostrando così l'uso di questo linguaggio come base su cui costruire linguaggi più evoluti) consentendo così una diretta sperimentazione delle proposte di integrazione.

a cura di E. Ciapessoni