Vol. XLV — Gennaio 1976 — N. 1

ALTA FREQUENZA

RIVISTA DI STUDI E RICERCA APPLICATA DELLA
ASSOCIAZIONE ELETTROTECNICA ED ELETTRONICA ITALIANA
SOTTO GLI AUSPICI DEL CONSIGLIO NAZIONALE DELLE RICERCHE
FONDATA DA GIANCARLO VALLAURI

Proprietaria ed Editrice: AEI - ASSOCIAZIONE ELETTROTECNICA ED ELETTRONICA ITALIANA.
Comitato per le pubblicazioni AEI: A. M. Angelini, F. Bianchi di Castelbianco, N. Faletti, E. Gatti, A. Gigli, P. Lombardi, P. Regoliosi, R. Sartori, G. Someda, F. Tedeschi, D. Tolomeo.
Direttore responsabile: E. Gatti.
Segretaria di Redazione: L. Tiné.
La rivista è pubblicata col concorso del Consiglio Nazionale delle Ricerche e della Fondazione Ugo Bordoni.

SOMMARIO

AVVERTENZE AGLI AUTORI — Gli Autori sono invitati a presentare i loro lavori che sono sottoposti all'esame della Redazione per la pubblicazione. I lavori devono essere originali e devono rientrare in una delle seguenti categorie:
a) *Articoli di rassegna critica e articoli di inquadramento destinati a progettisti e ricercatori.*
b) *Articoli scientifici o tecnici* relativi a studi e ricerche di tipo applicativo in campi avanzati dell'elettrotecnica e dell'elettronica.
c) *Lettere alla Redazione.* Comunicazioni critiche su lavori precedentemente pubblicati; brevi note scientifiche su ricerche compiute e sui risultati ottenuti, anche se preliminari; proposte di nuove ricerche.
d) *Dai Laboratori di ricerca e sviluppo.* Relazioni tecniche di interesse specialistico, da Industrie, Enti d'esercizio, Centri di ricerca o Università.
e) Relazioni di congressi scientifici e tecnici di interesse specialistico.
I manoscritti vanno inviati, in duplice copia, ad « Alta Frequenza », Viale Monza 259, 20126 Milano, e devono portare, per ognuno, nome, cognome, ente di appartenenza o indirizzo personale dell'Autore. Al fine del conferimento dei Premi AEI, agli Autori Soci dell'AEI è anche richiesta l'indicazione della data di nascita e della Sezione AEI di appartenenza. Per la presentazione formale dei manoscritti si consiglia di attenersi alle « Raccomandazioni agli Autori » che possono essere richieste alla Redazione e che vengono pubblicate ogni anno nel primo fascicolo in italiano della rivista.

L'IMPIEGO DELLA LOGICA MATEMATICA E DEI LINGUAGGI ORIENTATI AD OBIETTIVI NELLA FORMAZIONE DI PIANI PER ROBOT

P. CORTI (*) - G. GINI (*) - M. GINI (*) - E. PAGELLO (**)

Lo scopo di questo lavoro è quello di studiare ed affrontare alcuni problemi relativi alla manipolazione di oggetti da parte di un robot intelligente e di proporre alcune metodologie utili per ottenere le soluzioni di questi problemi.

L'esigenza di dotare un robot di autonome capacità deduttive è stata collegata a quella di disporre, sul calcolatore, di un linguaggio di deduzione. Le tecniche avanzate permettono di progettare un programma, scritto in tale linguaggio che rende il robot in grado di pianificare le sue azioni per conseguire un obiettivo all'interno di un suo «mondo» determinato e limitato.

Mediante la descrizione della configurazione iniziale del mondo, delle leggi in esso valide, e dello scopo da raggiungere, il programma genera il «piano» per il robot, trova cioè la sequenza di azioni primitive che rendono possibile passare dalla situazione iniziale a quella finale.

A tale sequenza può facilmente corrispondere la sequenza di azioni fisiche che il robot deve eseguire.

Si descrive un sistema deduttivo basato sulla logica dei predicati che permette di risolvere problemi di manipolazione.

Si mostra inoltre come questa capacità deduttiva possa essere conseguita, al di fuori di un sistema logico-formale, anche mediante l'uso di linguaggi particolari, orientati ad obiettivi.

Viene discussa infine la possibilità e la concreta convenienza di una effettiva realizzazione di sistemi di deduzione adatti a problemi reali di robotica.

1. - Introduzione: La generazione di strategie per il robot.

Durante gli ultimi anni le ricerche sulla robotica hanno fornito un punto di aggregazione di molte tecnologie dell'Intelligenza Artificiale.

Con il termine robot non intendiamo una macchina che esegue una sequenza di azioni programmata a priori, ma pensiamo a un meccanismo, controllato da calcolatore, in grado di interagire con il mondo che lo circonda in maniera autonoma e ragionevolmente intelligente.

Un robot quindi è una macchina complessa, dotata di organi sensori, per scoprire informazioni sul mondo circostante, di capacità deduttive, per progettare la soluzione di problemi che gli sono posti, e di organi mobili, per realizzare la soluzione trovata.

I problemi che la robotica presenta sono quindi molteplici, e vanno dallo studio di sistemi visivi efficienti alla progettazione di interfaccia fra il calcolatore e i dispositivi meccanici e sensori, alla realizzazione di sistemi dotati di capacità deduttiva.

Il nostro interesse è legato all'illustrazione e alla discussione delle esigenze e delle possibili realizzazioni di un tale sistema deduttivo.

L'esigenza di fornire il robot di una autonoma capacità deduttiva è correlata con la disponibilità, sul calcolatore che lo controlla, di un opportuno linguaggio di deduzione. Mediante tale linguaggio è possibile progettare un programma che consenta al robot di pianificare le sue azioni per raggiungere uno scopo assegnato [1].

Caratteristica comune di questi sistemi è quella di operare su una base dei dati in cui è memorizzato un modello del mondo cui far riferimento durante il processo di costruzione del piano. Tale modello del mondo, inoltre, deve essere tenuto aggiornato seguendo i cambiamenti che intervengono nel mondo reale.

I problemi della robotica richiedono modelli del mondo molto più complessi e generali di quelli necessari per la soluzione dei giochi e dei puzzles solitamente portati ad esempio nella letteratura di Intelligenza Artificiale. In tali semplici problemi infatti è sufficiente impiegare una lista o una matrice per rappresentare uno stato del problema.

Il modello del mondo necessario per risolvere problemi di robotica richiede invece un gran numero di dati e di relazioni per descrivere posizioni e caratteristiche degli oggetti, dell'ambiente o del robot stesso.

In genere la conoscenza sul mondo può essere divisa in due categorie:

(i) la conoscenza sul mondo in un dato istante o stato.

(ii) le informazioni che descrivono come il mondo può essere trasformato sulla base delle azioni del robot. Tali trasformazioni possono essere rappresentate da una collezione di operatori [2].

Un tale modello può essere ottenuto solo introducendo ipotesi restrittive e semplificazioni sul mondo reale, ma deve essere sufficientemente rispondente alla realtà esterna, in ogni momento della sua evoluzione. Infatti per il robot non è sufficiente costruire teoricamente la soluzione di un problema, ma è necessario eseguire le azioni che costruiscono effettivamente questa soluzione nel mondo reale.

Quindi, il compito richiesto al sistema deduttivo del robot è quello di gnerare, operando sul modello

(*) MP-AI Project - Istituto di Elettrotecnica ed Elettronica Politecnico di Milano.
(**) Laboratorio di Elettronica Industriale del C.N.R. - Padova.

del mondo, il *piano*, cioè la sequenza di azioni primitive che permettono lo transizione dalla situazione attuale del mondo ad una nuova, posta come obiettivo.

A tale piano, costruito sul modello del mondo, dovrà corrispondere una sequenza di azioni fisiche nel mondo reale.

La ricerca sulla robotica, iniziata verso il 1960, ha prodotto sistemi capaci di progettare ed eseguire, in modo intelligente, piani di azione basati sul modello interno del mondo. Tali sistemi sono stati realizzati allo S.R.I., alla Stanford University, al M.I.T., e a Edinburgo.

In essi sono in uso due approcci fondamentali alla programmazione.

Il primo di essi è caratterizzato dalla famiglia di programmi basati sull'uso della logica formale [3]. Questi sistemi sono molto generali e pensati per la programmazione di un robot in grado di affrontare problemi di ogni tipo; questa generalità va però a scapito dell'efficienza.

Di tali sistemi solo lo STRIPS [4] è impiegato per un robot sperimentale allo S.R.I.

Il secondo approccio, introdotto da Winograd [5] è sviluppato in seguito al M.I.T. da Fahlman [6] è basato sull'uso di linguaggi appositamente progettati e richiede la scelta di un dominio specifico in cui scrivere programmi che risolvono problemi in quell'ambito. Le leggi di inferenza sono specializzate ad operare la soluzione nel mondo fissato e questo permette di usare meglio l'informazione legata al dominio, e di guidare il processo in modo appropriato.

La scrittura delle procedure è facilitata dall'uso di linguaggi speciali, detti orientati ad obiettivi, basati sulla filosofia dell'approccio procedurale alla risoluzione dei problemi introdotto col PLANNER da Hewitt [7] [8].

2. - Applicazione della logica formale ai problemi di robotica.

Il primo esempio organico di sistema basato sull'uso della logica dei predicati per la risoluzione dei problemi è il QA3 di Green [3] che fu usato come base per le ricerche in questo settore. Anche se oggi non è più usato è necessario analizzarne le linee fondamentali per poter confrontare fra loro gli attuali sistemi.

Il QA3 si basa sull'uso di una procedura di prova, fondata sulla applicazione del principio di risoluzione di Robinson per il Calcolo dei Predicati del 1° Ordine.

Tale principio permette di inferire una nuova clausola logica da 2 clausole note al sistema, in particolare quindi di inferire nuove clausole dagli assiomi.

Questa procedura di prova si basa sulla proprietà di completezza del calcolo dei predicati, che afferma che ogni proposizione logicamente valida è anche deducibile dagli assiomi. In particolare essa tenta, dato un insieme S di assiomi e una formula W da dimostrare, di verificare la insoddisfacibilità dell'insieme di formule $S \cup (\sim W)$.

Il sistema di Green è stato applicato a molti domini semantici fra cui appunto quello della pianificazione delle azioni di un robot, problema che consiste nel costruire una sequenza di azioni che trasformano uno stato iniziale in uno stato finale assegnato.

Il sistema si basa sulla introduzione, nel formalismo logico di costruzione della prova, dei metodi di trasformazione di stato [2].

I fatti che descrivono il mondo in oggetto sono dati sotto forma di espressioni del Calcolo dei Predicati del 1° ordine.

Ad ogni istante il mondo in oggetto si trova in un determinato stato; per esempio, se il predicato

$$(1) \qquad P(x, s_0)$$

è vero, x gode della proprietà P nello stato s_0.

Mediante il compimento di azioni singole, o di sequenze di azioni, si ottengono cambiamenti di stato; una azione si può rappresentare come una funzione che applicata a uno stato costruisce un nuovo stato. Si ottiene per esempio la descrizione dell'effetto di una azione con il seguente assioma:

$$(2) \qquad (\forall\, s)(P(s) \supset Q(f(s)))$$

che indica che la azione in oggetto, espressa dalla funzione f, ha l'effetto di passare dalla situazione descritta dal predicato P nello stato s alla situazione descritta dal predicato Q nel nuovo stato $f(s)$.

Si può illustrare un esempio [3], nel quale viene descritta l'esistenza di un blocco (indicato con la lettera B) situato nella posizione p e trasportabile attraverso q in r sotto l'effetto di una azione descritta dalla funzione « push » che ha come argomento il blocco, le posizioni di partenza ed arrivo, e lo stato iniziale.

Si ottengono le seguenti espressioni logiche:

$$(3) \qquad At(B, p, s_0)$$

$$(4) \qquad (\forall s)(At(B, p, s) \supset At(B, q, \text{push}(B, p, q, s)))$$

$$(5) \qquad (\forall s)(At(B, q, s) \supset At(B, r, \text{push}(B, q, r, s)))$$

dove At è un predicato che afferma che B si trova nella posizione p nello stato s_0, e il termine push (B, x, y, s) caratterizza il nuovo stato in cui si trova il blocco sotto l'effetto della azione applicata nello stato s.

Se si inserisce una nuova espressione logica del tipo:

$$(6) \qquad \sim At(B, r, s), \text{Answer}(s)$$

e si applica una procedura di prova per determinare se è possibile portare il blocco dalla posizione p alla posizione r e quale sarà lo stato risultante del mondo, si ottiene la seguente catena di deduzioni:

da (6) e (5) posto in forma a clausole, si ricava:

$$(7) \qquad \sim At(B, q, s), \text{Answer}(\text{push}(B, q, r, s))$$

da (7) e (4) posto in forma a clausole, si ricava:

$$(8) \qquad \sim At(B, p, s), \text{Answer}(\text{push}(B, q, r, \text{push}(B, p, q, s)))$$

da (3) e (8) si ricava:

$$(9) \qquad \text{Answer}(\text{push}(B, q, r, \text{push}(B, p, q, s_0)))$$

Si è ottenuto cioè come risposta che è possibile compiere questa azione e lo stato risultante è ca-

ratterizzato dall'argomento del predicato Answer che compare in (9).

Green prevede l'esistenza di un ulteriore assioma da aggiungere al sistema visto che specifichi che tutti i blocchi diversi da quello in oggetto non vengono alterati dall'azione.

Se nel mondo quindi esistono altri blocchi, espressi da assiomi del tipo:

$$(10) \qquad At(B_i, m, s_0)$$

e vale l'ipotesi che l'azione push viene applicata al solo blocco B, viene aggiunto un ulteriore assioma:

$$(11) \qquad (\forall w, x, y, z, s)(At(w, x, s) \wedge \mathrm{Diff}(w, B) \supset At(w, x, \mathrm{push}(B, y, z, s)))$$

che deve essere attivato dopo ogni applicazione della funzione push per dedurre la nuova espressione degli assiomi (11).

Questo sistema presenta due caratteristiche negative. Infatti nel processo di deduzione coesistono due distinte procedure di ricerca, una nello spazio di tutti i possibili modelli del mondo, e l'altra nello spazio delle possibili soluzioni dell'obiettivo nell'ambito del modello scelto. Si ha quindi il sovrapporsi all'interno di una procedura di prova di due distinte ricerche guidate dal meccanismo di risoluzione del dimostratore di teoremi. Inoltre la particolare soluzione scelta per quello che viene chiamato il « frame problem », cioè il problema di determinare quali sono i blocchi che non vengono influenzati dall'esecuzione della sequenza di azioni, presenta molti inconvenienti. Il sistema deve infatti descrivere con assiomi speciali quelle relazioni che non sono influenzate dalla esecuzione della azione e, durante la procedura di prova, dopo ogni applicazione della funzione deve calcolare la posizione relativa di tutti i blocchi. Queste due impostazioni si traducono in un appesantimento combinatoriale dei passi della procedura di prova che diventa estremamente complessa rendendo poco efficiente il sistema deduttivo.

Poichè il compito del sistema è trovare una sequenza di operatori, che produrranno un modello del mondo in cui viene mostrato che la espressione logica esprimente l'obiettivo è vera, Fikes e Nilsson progettarono lo STRIPS [4] sulla base di una distinzione fra l'uso del dimostratore di teoremi all'interno di un modello del mondo fissato e l'uso dello stesso all'interno invece dello spazio dei modelli. Le modificazioni così introdotte ovviano agli inconvenienti citati, ma allontanano il sistema da una struttura puramente logica.

Gli operatori diventano così gli elementi in base ai quali si costruisce la soluzione, e la loro definizione viene resa più complessa di quanto non fossero le funzioni di cambiamento di stato in [3].

Ad ogni operatore vengono associati:

(i) il nome e i parametri

(ii) le condizioni di applicabilità dette « preconditions »

(iii) la lista di ciò che non è più vero e di ciò che si deve aggiungere, dette rispettivamente « delete list » e « add list ».

Con questo sistema, nel corso della procedura di prova, viene evidenziato il ruolo autonomo svolto dagli operatori, che sono resi molto potenti dalle proprietà (ii) e (iii). Inoltre, per evitare di calcolare dopo ogni nuova applicazione di un operatore la condizione di ogni oggetto del mondo, si ricorre a nominare solo quelle relazioni che sono influenzate dall'operatore messo in gioco, assumendo che le relazioni non nominate rimangano valide e risolvendo così brillantemente in pratica il frame problem. Questo compito è affidato alle delete e add lista, mentre il sistema si presenta come l'attivazione di una serie di processi di verifica delle precondizioni, cioè di obiettivi locali, che permettono di conseguire l'obiettivo principale.

La ricerca di soddisfare l'obiettivo principale genera un processo di attivazione di obiettivi secondari da conseguire, che traducono la soluzione del problema principale in un insieme di sottoproblemi [2].

Una recente proposta [9] suggerisce di utilizzare i risultati ottenuti nel produrre un piano particolare mediante una generalizzazione che ne consenta un successivo impiego.

3. - Linguaggi orientati ad obiettivi e controllo di un robot.

La capacità fondamentale richiesta al programma che controlla un robot è, come si è già illustrato, quella di compiere deduzioni.

Per deduzione, nella logica usuale, intendiamo la legge che ci assicura che, dato « A implica B » e data la verità di « A », anche « B » è vero.

Nella logica formale questa regola permette, dato un insieme di buone formule, di ricavarne altre valide o, in altre parole, di dimostrare che alcune buone formule sono valide.

L'implementazione dei linguaggi orientati ad obiettivi ha permesso la realizzazione di capacità deduttive automatiche senza ricorrere ad un sistema logico-formale.

Il primo di questi linguaggi, il PLANNER, è stato progettato da Hewitt [7] e ne è stato implementato un sottoinsieme, il MICROPLANNER [8]. E' a quest'ultimo linguaggio che faremo riferimento in seguito.

In esso l'aspetto deduttivo è realizzato mediante opportune parti del programma, dette « teoremi », che contengono le leggi e le relazioni valide nel mondo in esame.

Ad esempio, il fatto che un oggetto U possa essere spostato dalla posizione W alla posizione V è espresso dal seguente teorema:

```
(PUT' SPOSTA' THEOREM' (THCONSE (U V W))
    (ON (THV U) (THV V))
    (THNOT (EQUAL (THV U) (THV V)))
    (THNOT (THGOAL ((THV U) LOCUS)))
    (THGOAL ((THV V) TOP))
    (THGOAL (ON (THV U) (THV W)))
    (THERASE (ON (THV U) (THV W)))
    (THASSERT (ON (THV U) (THV V))) ))
```

In esso si controlla che l'oggetto e la posizione siano distinti, che la posizione V sia libera, e si cambia quindi la descrizione del mondo asserendo che ora U si trova nella posizione V e non più in W.

I teoremi di questo tipo, detti teoremi di tipo consequent, sono caratterizzati da un « pattern » che esprime il risultato del teorema,

(ON (THV U)(THV V))

e da una serie di passi, detta corpo del teorema.

Il teorema è scritto in modo che « il corpo del teorema implica il pattern »; se vogliamo dimostrare la verità del pattern dobbiamo dimostrare, cioè eseguire con successo, il corpo del teorema. Quest'ultimo infatti è costituito da una serie di istruzioni, la cui valutazione dà come risultato « successo » o « fallimento ». La possibilità di compiere lunghe catene di deduzioni è assicurata dal fatto che queste istruzioni possono, a loro volta, richiedere l'attivazione di altri teoremi.

La conoscenza del mondo, su cui operare le deduzioni, è espressa in un insieme di « asserzioni ».

Ad esempio, l'esistenza di un blocco A nella posizione z è espressa dalle seguenti asserzioni:

(THASSERT (A BLOCK))
(THASSERT (ON A Z))
(THASSERT (Z LOCUS))

ed in modo analogo può essere definita l'esistenza di posizioni libere:

(THASSERT (X LOCUS)
(THASSERT (X TOP))

Sia i teoremi che le asserzioni, memorizzate in una « base dei dati », hanno un significato dichiarativo. Nel momento in cui viene assegnato il problema da risolvere, espresso sotto forma di « goal » da raggiungere, ad esempio:

(THGOAL (ON A X)) (THTBF THTRUE)

il sistema prova a soddisfare la richiesta cercando se esiste una asserzione che risolve il problema. Se questa situazione non si verifica, viene chiamato e attivato un teorema che sembri « adatto » al caso in esame. Il teorema assume quindi un significato imperativo, cioè rappresenta azioni da eseguire.

In questo caso, alla variabile *U* viene legato il blocco A, alla posizione *V* il posto X. I THGOAL sono verificati dalle asserzioni iniziali, e quindi il teorema ha successo.

Il metodo di soluzione così realizzato è quello comunemente detto « top-down ». La deduzione cioè non parte dalle asserzioni cercando di dedurne tutte le possibili conseguenze, ma parte invece dal goal da raggiungere e cerca, in passi successivi, di spezzarlo in sottogoals la cui soluzione sia direttamente ottenibile dalle asserzioni iniziali.

Uno degli aspetti più importanti dei linguaggi orientati ad obiettivi è il fatto che i teoremi non vengono chiamati in base al loro nome (come per le ordinarie procedure), ma in base a ciò che essi sono in grado di dimostrare, che è espresso nel pattern associato ad ogni teorema.

Esiste un meccanismo di « pattern matching » che permette di legare le variabili presenti nel pattern del teorema a quello del goal chiamante, o di ritrovare asserzioni dandone una descrizione incompleta.

Ci possono essere diverse asserzioni o diversi teoremi candidati per la ricerca della soluzione. Ogni volta che si presentano diverse alternative se ne sceglie una e la si esplora fino in fondo, mantenendo la capacità di respingere la scelta e le sue conseguenze e di tornare al punto di decisione per fare una scelta diversa. Tale struttura di controllo è automatica e viene detta « backtracking ». C'è anche la possibilità di guidare il processo di ricerca della soluzione dando consigli sui teoremi da usare, indicando se la ricerca va effettuata solo fra le asserzioni, e così via.

Il MICROPLANNER permette quindi la implementazione di programmi non deterministici.

La possibilità di scegliere strade che portano a un fallimento e che quindi devono essere annullate non costituisce un elemento negativo per le applicazioni alla robotica. Infatti tali fallimenti avvengono, nella fase di ricerca della soluzione, sulla rappresentazione interna al calcolatore, mentre al braccio del robot viene fornita da eseguire solo la sequenza corretta di azioni.

La scelta di utilizzare linguaggi orientati ad obiettivi per i problemi della robotica nasce dalle motivazioni che hanno portato, già da alcuni anni, al superamento di linguaggi come il LISP, con struttura di controllo solo ricorsiva.

Infatti, pur essendo il LISP sufficiente da un punto di vista computazionale, si è sentita l'esigenza di avere a disposizione meccanismi addizionali per compiti specifici, come quello di annullare scelte sbagliate, di stabilire una gerarchia di goal e sottogoals, di mantenere diversi modelli del mondo.

Questa esigenza ha portato alla creazione di una serie di meccanismi di controllo standard e di strutture dati adeguate alla maggior parte dei programmi di risoluzione automatica dei problemi, pur lasciando una sufficiente generalità per poter operare su svariati domini.

4. I sistemi di deduzione lineare e il problema dei 3 blocchi.

Abbiamo visto che sia lo STRIPS che i linguaggi orientati ad obiettivi operano una scomposizione del problema in sottoproblemi.

Il più semplice approccio per risolvere un problema ridotto a sottoproblemi è di costruire una struttura di controllo di attivazione di tipo lineare, tale cioè che la risoluzione di due sottoproblemi avvenga in modo indipendente. Allora l'obiettivo « $G_0 \cup G_1$ » può essere risolto a partire da uno stato iniziale, mediante una sequenza di operatori applicati allo stato iniziale, che risolvano G_0 e successivamente applicando un'altra serie di operatori allo stato risultante precedente, che risolvano G_1. Nel caso di fallimento si potrà mutare l'ordine dei sottoproblemi cercando di risolvere prima G_1 e poi G_0.

Deve esistere perciò nel sistema una struttura di controllo che in base alla descrizione degli operatori data ne utilizzi le relative proprietà al fine di conseguire i sottoobiettivi in oggetto.

Si può studiare l'effetto di un tale sistema sul problema di manipolazione dei tre blocchi, come illustrato in [10] e [11]; secondo Tate è proprio in tale esempio che sono evidenti le limitazioni degli attuali sistemi lineari.

Il mondo dei blocchi è l'ideale per gli studi di robotica, perchè è un mondo chiuso e abbastanza semplice, ma fornisce problemi interessanti. Bastano pochi concetti per poter organizzare i programmi senza dover affrontare la difficoltà di una rappresentazione globale del mondo; inoltre le ipotesi restrittive che devono essere introdotte non nuocciono alla costruzione di una soluzione valida.

Dato un mondo costituito da tre blocchi, A, B, C posati su di un tavolo t tale che i tre blocchi possono essere manipolabili e sovrapponibili, si consideri lo stato rappresentato in fig. 1, da cui si vuole ottenere quello rappresentato in fig. 2.

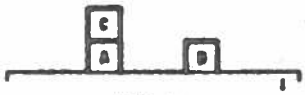

Fig. 1

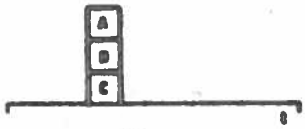

Fig. 2

Si suppone l'esistenza di un operatore pickup (u, x, y, s) che esprime l'azione di portare il blocco u da x sopra y, che esige il soddisfarsi delle « preconditions »:

(12) $\quad (\exists u, x, y)(\text{on}\,(u, x) \wedge \text{clear}\,(y))$

e le cui add e delete lists sono rispettivamente:

(13) $\quad \text{on}\,(u, y) \wedge \text{clear}\,(x)$

(14) $\quad \text{clear}\,(y) \wedge \text{on}\,(u, x)$,

e si suppone di descrivere il conseguimento dello stato finale come la risoluzione della congiunzione dei due sottoproblemi

(15) $\quad \text{on}\,(A, B) \wedge \text{on}\,(B, C)$

Un sistema lineare procede alla ricerca della soluzione attraverso i seguenti passi: per conseguire l'obiettivo $\text{on}\,(A, B)$ tramite l'operazione pickup, si ottengono successivamente le configurazioni di fig. 3 e fig. 4.

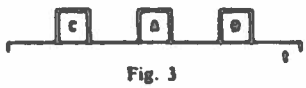

Fig. 3

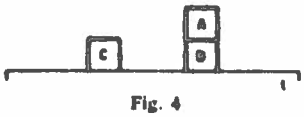

Fig. 4

Quindi per conseguire il secondo obiettivo $\text{on}\,(B, C)$, si propone di applicare l'operatore pickup così da ottenere la configurazione di fig. 5, ma questa distrugge un obiettivo già conseguito. Per la linearità del sistema viene commutato l'ordine e si cercano di conseguire successivamente gli obiettivi:

(16) $\quad \text{on}\,(B, C) \wedge \text{on}\,(A, B)$

Il primo viene conseguito, tramite l'operatore pickup, ottenendo la configurazione (fig. 6) a partire dalla configurazione iniziale (fig. 1), mentre per conseguire il secondo si deve passare attraverso la configurazione di fig. 7.

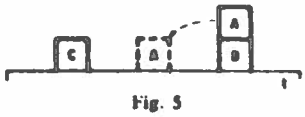

Fig. 5

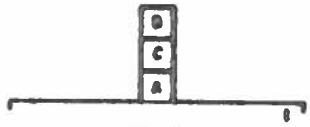

Fig. 6

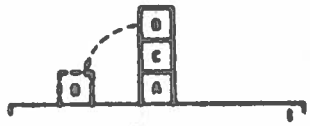

Fig. 7

Ma questa distrugge l'obiettivo precedentemente conseguito.

Perciò il sistema lineare proposto o conclude falsamente che il problema è insolubile, oppure, se la sua struttura di controllo è abbastanza flessibile da permettergli di violare le condizioni restrittive di non distruggere l'obiettivo precedente, ottiene una soluzione non ottima (nel senso del numero delle operazioni), mentre la sequenza ottima di azioni sarebbe la seguente:

(17) $\quad \text{pickup}\,(C, A, t)$; $\text{pickup}\,(B, t, C)$; $\text{pickup}\,(A, t, B)$.

Il fatto quindi che la soluzione di un sottoproblema si intrecci con quella dell'altro rende questa classe di problemi di manipolazione sui blocchi di notevole interesse per la determinazione della complessità dei sistemi di risoluzione.

5. - L'approccio strettamente logico.

Si può affrontare il problema della manipolazione dei blocchi tornando allo spirito originario del sistema di Green, e cioè ad una descrizione del sistema compatta in forma di espressioni del calcolo dei predicati, e introducendo in espressioni logiche lo spirito del concetto di operatori che Fikes ed Nilsson avevano usato con una notazione che rompeva l'omogeneità fra descrizione del mondo e sistema di trasformazione degli stati.

Lo strumento per ottenere tale descrizione compatta è l'interpretazione procedurale delle clausole di Horn, fornita da Kolwalsky [12].

In tale interpretazione:

(i) Le clausole, del tipo

(18) $\quad B \leftarrow A_1, \ldots, A_n$

contenenti le variabili $x_1, \ldots, x_m$, si leggono:

per tutti gli $x_1 \ldots x_m$
B è implicato da A_1 e A_2 e $\ldots$ e A_n.

(ii) Le clausole del tipo

(19) $$B \leftarrow$$

corrispondono ad asserzioni, e si leggono:

per tutti gli $x_1 \dots x_m$
B

(iii) Le clausole del tipo

(20) $$\leftarrow A_1, A_2 \dots A_n$$

corrispondono a goals, o clausole finali e si leggono:

per nessun $x_1 \dots x_m$
A_1 e A_2 e $\dots$ e A_n

L'uso di clausole del tipo

$$B_1, B_2, \dots, B_k \leftarrow A_1, \dots, A_n$$

che si leggono:

per tutti gli $x_1 \dots x_m$
B_1 o $\dots$ B_k è implicato da A_1 e $\dots$ A_n

può essere introdotto per permettere la generazione di piani condizionali.

Queste clausole non hanno una chiara interpretazione procedurale e spesso conducono a problemi di sottodeterminazione dell'uscita, che possono essere inquadrati in uno studio più generale dei processi di computazione e deduzione.

E' stato inoltre mostrato, in generale, che le clausole di Horn sono adeguate a definire tutte le relazioni esprimibili nel calcolo dei predicati del primo ordine.

In tale sistema formale, si può ancora assumere che a ciascuna azione che il robot compie siano associate:

(i) delle condizioni iniziali che devono essere vere in uno stato affinchè l'azione sia ad esso applicabile;

(ii) una lista di nuove asserzioni che assumono il valore di verità nello stato ottenuto applicando l'azione;

(iii) una lista delle vecchie asserzioni che non sono più vere nel nuovo stato in seguito all'azione compiuta.

Il problema dei tre blocchi perciò può essere compattamente descritto dal seguente insieme di proposizioni logiche, che esprimono sia le relazioni fra gli oggetti del mondo, sia le proprietà degli operatori a loro applicati.

La descrizione è analoga a quella proposta da Kowalski [12].

— Assioma di descrizione della proprietà del tavolo t

(21) $$\text{Holds}(\text{clear}(t), s) \leftarrow$$

— Assiomi di trasformazione dello spazio degli stati

(22) $$\text{Poss}(p(u, x, y, s) \leftarrow \text{Pact}(p(u, x, y, s)), \text{Poss}(s)$$

(23) $$\text{Poss}(0) \leftarrow$$

— Assiomi di descrizione dello stato iniziale 0

(24) $$\text{Holds}(\text{on}(C, A), 0) \leftarrow$$

(25) $$\text{Holds}(\text{on}(A, t), 0) \leftarrow$$

(26) $$\text{Holds}(\text{on}(B, t), 0) \leftarrow$$

(27) $$\text{Holds}(\text{clear}(C), 0) \leftarrow$$

(28) $$\text{Holds}(\text{clear}(B), 0) \leftarrow$$

— Assioma per le condizioni a cui deve sottostare l'operatore p

(29) $$\text{Pact}(p(u, x, y, s)) \leftarrow \text{Holds}(\text{clear}(u), s), \text{Holds}(\text{clear}(y), s), \text{Holds}(\text{on}(u, x), s)$$

— Add list

(30) $$\text{Holds}(\text{on}(u, y), p(u, x, y, s)) \leftarrow$$

(31) $$\text{Holds}(\text{clear}(x), p(u, x, y, s)) \leftarrow$$

— Delete list

(32) $$\text{Holds}(w, p(u, x, y, s)) \leftarrow \leftarrow \text{Holds}(w, s), \text{Diff}(w, \text{clear}(t)), \text{Diff}(w, \text{on}(u, x)), \text{Diff}(w, \text{clear}(y))$$

che si interpretano nel seguente modo:

on (x, y) è un termine che esprime la proprietà che x è sopra y; clear (m) è un termine che esprime la proprietà che sopra m si può posare un cubo, intendendo con l'assioma (21) che sopra il tavolo si possono posare un numero illimitato di cubi; Holds è un simbolo predicativo che afferma la proprietà che i termini precedenti valgono nello stato indicato dal 2° argomento; $p(u, x, y, s)$ è un termine che esprime, se dato come 2° argomento di Holds o come argomento di Poss, lo stato in cui si trova il sistema dopo l'azione pickup (u, x, y); Poss è un simbolo predicativo che afferma la possibilità dello stato suo argomento; Pact è un simbolo predicativo che afferma la possibilità di compiere l'azione pickup (u, x, y) nello stato s; Diff è un predicato che prova l'uguaglianza degli argomenti.

L'assioma (32) richiede la presenza di un insieme, eventualmente infinito, di clausole del tipo:

$$\text{Diff}(s, t) \leftarrow$$

per ogni coppia di termini s e t senza variabili e non unificabili. Si può anche pensare che il predicato Diff sia implementato a livello del sistema usato, e che venga computato opportunamente quando si trova nella ipotesi di una clausola.

L'assioma (22) permette di considerare le trasformazioni subite dallo stato per mezzo dell'azione degli operatori tramite le preconditions (29) e le add e delete lists (30) (31) (32).

La esecuzione di questo assioma in maniera « top-down », che parte cioè dalla clausola finale e cerca di generare altri goals, o in maniera « bottom up », che parte cioè dalle asserzioni e cerca di generare nuove asserzioni, permette di ottenere due diversi spazi di ricerca della soluzione ottima, e divide il problema in diversi approcci. Se si esprime il conseguimento dello stato finale rappresentato in fig. 2 con la espressione:

(33) $$\leftarrow \text{Holds}(\text{on}(A, B), s), \text{Holds}(\text{on}(B, C), s), \text{Holds}(\text{on}(C, t), s), \text{Poss}(s)$$

si può attivare un processo di deduzione logica basato sul principio di risoluzione che esegue le clausole (21)/(33) in maniera top-down o bottom-up, otte-

nendo una diversa interpretazione degli assiomi (22) e (32).

Una interpretazione bottom-up dell'assioma (22) è usata per derivare che un nuovo stato s' è possibile data l'assunzione che il vecchio stato s sia possibile e che l'azione p possa essere applicata nello stato s così da esprimere s' come $p(u, x, y, s)$.

Una applicazione ripetuta di questo tipo fa cominciare la deduzione dallo stato iniziale, applica l'azione per produrre nuovi stati da vecchi e termina quando essa genera uno stato che soddisfa la descrizione dello stato finale.

Il suo albero di ricerca sarà espresso dal diagramma rappresentato in fig. 8.

stato s è possibile e che l'assioma p può essere applicato nello stato s. Una applicazione ripetuta di questo tipo dell'assioma (22) fa cominciare la deduzione dall'attivazione di (33), come illustrato in fig. 9, il che corrisponde logicamente a provare l'inconsistenza della negazione del teorema in oggetto nell'ambito degli assiomi (21)/(32); la deduzione termina quando si è trovata una descrizione di un nuovo stato che è soddisfatta dalle condizioni (24)/(28) relative allo stato iniziale.

Per un problema come quello della manipolazione di blocchi, sembra che una analisi completa nello spazio di ricerca corrispondente ad una attivazione

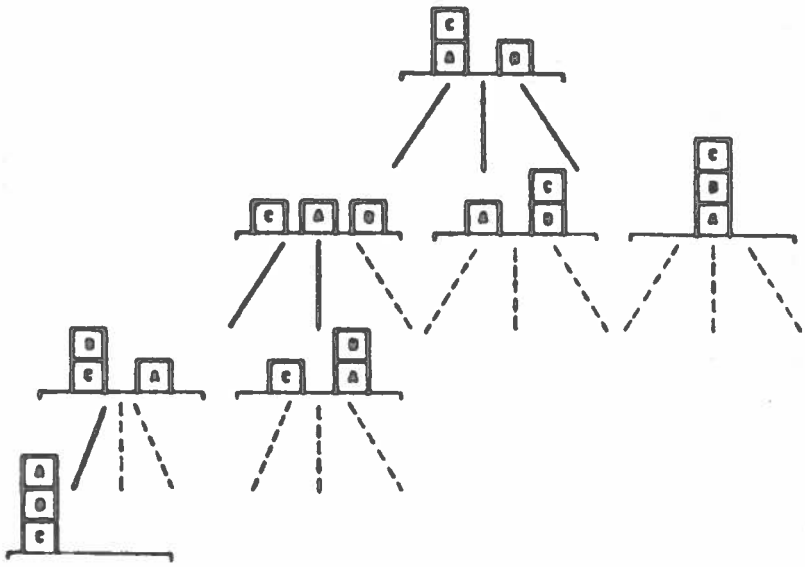

Fig. 8

Una interpretazione top-down invece dello stesso assioma è usata per ridurre il problema di verificare che un nuovo stato s', espresso come $p(u, x, y, s)$ è possibile, ai sottoproblemi di mostrare che il vecchio

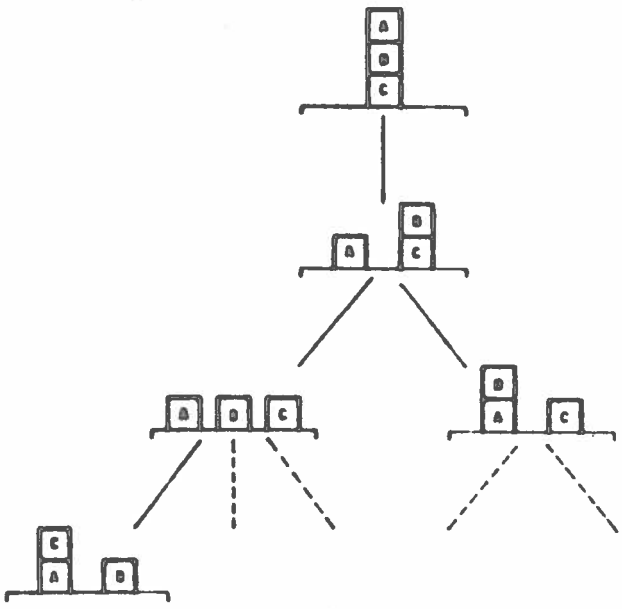

Fig. 9

di tipo top-down risolva il problema più efficacemente. Infatti la attivazione di tipo bottom-up dell'assioma (32), che corrisponde al «frame problem» citato, in un mondo di oggetti appena superiore di numero a quello dato condurrebbe ad una esplosione esponenziale del numero di espressioni logiche da derivare ad ogni nuovo stato generato. Una interpretazione di tipo top-down dello stesso assioma riduce lo spazio di ricerca, rendendo necessarie solo le derivazioni relative al cammino deduttivo legato alla verifica dello stato finale. Le difficoltà di una tale attivazione sorgono invece relativamenete alla introduzione di un numero sempre più grande di variabili nei successivi sottoproblemi generati, che deve essere ristretto in qualche modo, potendosi usare le condizioni dello stato iniziale solo nel passo finale del processo di deduzione, sebbene queste condizioni possano guidare euristicamente la ricerca del cammino risolutivo nella strategia di analisi dell'albero delle possibilità, suggerendo delle procedure selettive.

In appendice (app. A), si riporta il cammino ottimo appartenente allo spazio di ricerca generato da una attivazione puramente top-down della espressione (33); esso corrisponde alla soluzione ottima indicata alla fine della sezione precedente.

6. - L'uso del MICROPLANNER.

L'uso di un linguaggio orientato ad obiettivi richiede di analizzare il problema in modo un po' diverso da quello illustrato in precedenza.

Anche in questo caso il mondo viene simulato mediante un insieme di asserzioni.

Lo stato iniziale, rappresentato in fig. 1, è definito mediante il seguente insieme di asserzioni:

```
(THASSERT (A BLOCK))
(THASSERT (B BLOCK))
(THASSERT (C BLOCK))
```

che definiscono l'esistenza di 3 cubi,

```
(THASSERT (ON C A))
(THASSERT (ON B TAB))
(THASSERT (ON A TAB)
```

che definiscono la posizione dei blocchi sulla tavola TAB,

```
(THASSERT (B TOP))
(THASSERT (C TOP))
(THASSERT (TAB TOP))
```

che indicano quali sono gli oggetti su cui non è appoggiato altro, e che sul tavolo si può sempre appoggiare qualcosa.

Le leggi che regolano i possibili movimenti degli oggetti, sono espresse sotto forma di teoremi.

Il teorema MOVE

```
(PUT 'MOVE 'THEOREM '(THCONSE (U V W)
      (ON (THV U) (THV V))
      (THNOT (EQUAL (THV U) (THV V)))
      (THNOT (EQUAL (THV U) 'TAB))
      (THGOAL (ON (THV U) (THV W)))
A     (THOR (THGOAL ((THV V) TOP))
            (THAND (THGOAL (CLEAR (THV V)
            NOT ON (THV V))
            (THUSE CLEAR)) (THGO A)))
B     (THOR (THGOAL ((THV U) TOP))
            (THAND (THGOAL (CLEAR (THV U)
            NOT ON (THV V))
            (THUSE CLEAR)) (THGO B)))
            (THERASE (ON (THV U) (THV W)))
            (THCOND ((THNOT (EQUAL (THV V)
            TAB)) (THERASE
            ((THV V) TOP))) (T THSUCCEED))
            (THASSERT (ON (THV U) (THV V)))
            (THCOND ((THNOT (EQUAL
            (THV W) 'TAB))
            (THASSERT ((THV W) TOP)))
            (T THSUCCEED))))
            (THBKPT (THV U) 'IS 'POSED 'ON
            (THV V))
```

permette di spostare un blocco U sopra V. Il teorema esegue un controllo per assicurarsi che U e V siano distinti e che U non sia il tavolo.

Occorre poi determinare il blocco o la posizione W su cui si trova U prima di iniziare il movimento, e controllare che U e V non abbiano sopra altri oggetti. In questo caso è necessario liberare il blocco in esame chiamando un teorema che tolga gli oggetti sovrastanti e li metta in una posizione diversa da V.

Eseguite queste operazioni occorre aggiornare la conoscenza del mondo mediante cancellazioni e nuove asserzioni.

Un esempio di uso di questo teorema è illustrato in fig. 10.

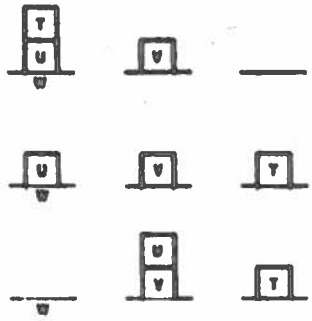

Fig. 10

Il teorema CLEAR

```
(PUT 'CLEAR 'THEOREM '(THCONSE (X Y Z V)
      (CLEAR (THV X) NOT ON (THV V))
      (THGOAL (ON (THV Z) (THV X)))
      (THOR (THAND (THGOAL ((THV Z) TOP))
                   (THGOAL ((THV Y) TOP))
                   (THNOT (EQUAL (THV Y)
                   (THV V)))
                   (THGOAL (ON (THV Z)
                   (THV Y)) (THUSE MOVE)))
            (THGOAL (CLEAR (THV Z) NOT ON
                   (THV V)) (THUSE CLEAR)))))
```

permette di togliere i blocchi sovrastanti un blocco determinato, spostandoli in una posizione qualsiasi purchè distinta da quella che serve per la costruzione successiva.

Il teorema è di tipo ricorsivo, cioè controlla se sopra l'oggetto X c'è un solo oggetto Z e in tal caso lo sposta. Altrimenti il teorema viene richiamato per liberare l'oggetto che sta sopra X, cioè Z, fino ad arrivare a quello più in alto di tutti.

Il sistema è attivato assegnando lo stato finale richiesto sotto forma di goal da raggiungere:

```
(THAND (THGOAL (ON C TAB) (THUSE MOVE))
       (THGOAL (ON B C) (THUSE MOVE))
       (THGOAL (ON A B) (THUSE MOVE)))
```

La soluzione ottenuta è la soluzione ottima:

A viene messo sul tavolo
B viene posto su C
A viene posto su B

In app. B viene riportato il tabulato relativo al problema enunciato, da cui è possibile seguire i legami realizzati fra le variabili e l'attivazione successiva dei sottogoals che rende possibile il raggiungimento della soluzione. L'uso di un particolare funzione permette infatti di ottenere, oltre al risultato, anche la traccia della valutazione.

In realtà il risultato di un programma MICROPLANNER è di tipo binario, cioè il programma « ha successo » se il goal è raggiungibile, e « fallisce » altrimenti. Con l'uso di opportuni indicatori è possibile ottenere, in maniera molto semplice, la successione dei passi necessari per giungere al successo, cioè quella che viene comunemente indicata come soluzione.

La scelta del MICROPLANNER è motivata dalle

caratteristiche del linguaggio, progettato appositamente, e dalla sua attuale disponibilità [13].

In realtà possono essere facilmente individuati gli aspetti in cui il MICROPLANNER non risulta del tutto soddisfacente.

Primo fra tutti, la rigidità della strategia di ricerca, che, pur rendendo la programmazione semplice, non consente la esplorazione di più rami in parallelo.

Il meccanismo di «pattern matching» inoltre è abbastanza elementare, e non sempre può essere guidato come sarebbe desiderabile, soprattutto quando si agisce sulle asserzioni.

Infine è molto oneroso mantenere contemporaneamente diversi modelli del mondo, e questo scoraggia uso di strategie in cui si debba far riferimento a stati precedenti.

7. - Confronto fra i sistemi esistenti e conclusioni.

Per problemi di manipolazione di strutture sono state implementate sul calcolatore diverse soluzioni che cercano di conseguire una soluzione ottima all'interno di sistemi di tipo logico formale allinando delle strategie di ricerca, oppure si basano su sistemi del tutto euristici.

Il «robot planner» di Sikklossy e Dreussi [14] abbandona completamente i sistemi analizzati e usa un insieme di procedure euristiche, risultando di gran lunga il più veloce sistema di elaborazioni di sequenze di azioni per un robot, ma non prestandosi ad alcuna teorizzazione.

Fahlman [6] ha scritto recentemente un programma in Conniver, per la costruzione di strutture complesse a partire da semplici blocchi. Progettato con una struttura di controllo di tipo euristico usa un assieme di tecniche sofisticate nella pianificazione delle azioni, come l'incorporazione delle strutture preesistenti nel progetto finale, il preassemblaggio di sottostrutture mobili.

Il sistema proposto da Sussman [11] propone una analisi ad apprendimento dei problemi di manipolazione che impari dalla generazione di sottoproblemi a non ripetere più gli errori che eventualmente si siano verificati, sfruttando quindi l'accadere di un errore concettuale ai fini del miglioramento della propria abilità.

Warren ha invece recentemente pubblicato [15] alcuni risultati molto interessanti del suo sistema WARPLAN. Questo è una felice combinazione dell'uso degli operatori, come sono dati nello STRIPS, con una struttura di controllo formalizzata in forma di espressioni logiche del Calcolo dei predicati. Il WARPLAN estende leggermente lo STRIPS per conseguire soluzioni ottime di molti problemi. Il WARPLAN risulta essere un dimostratore automatico di teoremi che agisce come struttura di controllo per generare i piani. Inoltre è programmato in PROLOG [16], un linguaggio rigorosamente basato sul calcolo dei predicati, il cui interprete è un dimostratore automatico di teoremi. Nel WARPLAN perciò si aggiunge agli assiomi (21), (24)/(33), invece dell'assioma di trasformazione di stato, una serie di espressioni logiche consistenti, orientate verso la semantica del dominio coinvolto, che danno al corso della procedura di prova un comportamento simile a quello dello STRIPS.

Una ulteriore estensione nella direzione dei sistemi tipo STRIPS è proposta da Sacerdoti [17] col sistema ABSTRIPS, che migliora il processo di ricerca mediante l'introduzione di un ordine gerarchico nelle informazioni presenti.

Un sistema come quello che è stato descritto nella sezione 5 è invece più simile a quello di Green in quanto richiede una strategia di prova strutturalmente di 1° ordine sulla quale poi si può innestare una euristica di tipo semantico. Un tale sistema quindi si presenta più compatto da un punto di vista formale. Quanto poi al conseguimento di quella soluzione ottima, che si è riportata in app. A (ottenendola in modo intuitivo) è interessante il lavoro di Kowalski [12], [18] che propone di usare o procedure rigorosamente top-down, oppure una intelligente combinazione di attivazioni «top-down» con attivazioni «bottom-up», che si combinino in un grafo di risoluzione chiamato «connection graph». Si ottiene così una compattezza nella struttura sintattica del sistema che può essere descritto come un dimostratore automatico di teoremi.

Il sistema presentato nella sez. 6 permette una rappresentazione abbastanza compatta e leggibile del problema. La soluzione viene ottenuta in maniera top-down con una buona efficienza nonostante il limite del backtracking automatico.

Il sistema è molto flessibile nel senso che è possibile, in maniera semplice, aggiungere nuove funzioni che rispondano a particolari necessità. Ad esempio se la soluzione di un problema espresso mediante diversi goals non è raggiungibile cercando di soddisfare i goals nell'ordine assegnato, si può aggiungere una funzione che determini tutte le diverse combinazioni dei goals stessi e vada a provarle.

Da quanto illustrato nelle sezioni precedenti è quindi possibile concludere che l'approccio logico è auspicabile in ogni problema in cui è necessaria una rigorosa descrizione logica, indipendente dal comportamento del calcolatore.

La logica infatti, secondo [12], viene intesa come linguaggio di programmazione, ad alto livello, non deterministico, la cui semantica è indipendente dalla macchina.

Usando poi in modo opportuno le strategie di ricerca si evita un eccessivo appesantimento computazionale.

L'approccio dei linguaggi orientati ad obiettivi, meno rigoroso ma più flessibile, è invece auspicabile probabilmente per la pratica della programmazione. L'uso infatti di linguaggi di questo tipo, soprattutto nelle versioni più recenti e evolute [19], [20], [21], [22], consente di introdurre caratteristiche delle logiche di ordine superiore, che rendono più espressivo e potente il sistema, senza un eccessivo appesantimento della ricerca. Tali linguaggi permettono inoltre tecniche di programmazione più sofisticate, mettendo a disposizione un insieme abbastanza ricco di funzioni o di strategie di controllo.

Si può infine prospettare l'interesse di questo approccio anche per alcune applicazioni alla robotica industriale; su alcuni di tali temi, legati all'uscita da situazioni di emergenza, è ora in atto una attività di ricerca presso il Progetto di Intelligenza Artificiale del Politecnico di Milano [23].

Appendice 1.

Refutazione top-down, ottenuta partendo da (33).

⟵ (22)	① Poss (s) ② Holds (on (A, B), s), ③ Holds (on (B, C) s), ④ Holds (on (C, t), s)
(30)	
(32)	
(32)	
⟵ (29)	① Pact (p (A, x, B, s')), ② Poss (s'), Holds (on (B, C), s'), Holds (on (C, t), s') (*)
(22)	
⟵ (30)	① Holds (clear (A), s') ④ Holds (clear (B), s'), ⑤ Holds (on (A, x), s'), Pact (p (u, x', y, s")), Poss (s"), ① Holds (on (B, C), s'), ② Holds (on (C, t), s')
(32)	
(32)	
(32)	
(32)	
⟵ (29)	Holds (clear (A), s"), Holds (clear (B), s"), Holds (on (A, x), s"), ① Pact (p (B, x', C, s")), ② Poss (s"), ③ Holds (on (C, t), s")
(22)	
(30)	
(23)	
(29)	
(27)	
(21)	
(24)	
⟵ (30)	① Holds clear (A), p (C, A, t, 0)), ② Holds (clear (B), s'"), Holds (on (A, x), s'"), Holds (clear (C), s'"), Holds (on (B, x'), s'")
(32)	
⟵ (32)	Holds (clear (B), 0), ① Holds (on (A, x), p (C, A, t, 0)), ② Holds (clear (C), p (C, A, t, 0)), ③ Holds (on (B, x'), p (C, A, t, 0))
(32)	
(32)	
⟵ (28)	① Holds (clear (B), 0), ② Holds (on (A, x), 0), ③ Holds (clear (C), 0), ④ Holds (on (B, x), 0)
(25)	
(27)	
(26)	
□	

(*) L'applicazione dell'assioma (32) introduce atomi del tipo *Diff*. (w, x). Essi vengono unificati pensando di avere a disposizione un insieme infinito di clausole per tutte le coppie di termini che non sono unificabili.

Appendice 2.

Risoluzione del problema dei 3 blocchi in MICRO-PLANNER.

```
(THAND (THGOAL (ON C TAB)(THUSE MOVE))
       (THGOAL (ON B C)(THUSE MOVE))
       (THGOAL (ON A B)(THUSE MOVE)))
>GOAL 01: (ON C TAB)
  >(THCONSE MOVE: (ON C (ANY X))
    >GOAL 02: (ON C (ANY X))
    <02 SUCCEEDED: ((ON C A))
    >GOAL 03: (CLR TOP)
    <03 SUCCEEDED: ((CLR TOP))
    >GOAL 04: (C TOP)
    <04 SUCCEEDED: ((C TOP))
    >PART 05: C IS PUSHED ON TAB
    >(ERASING 02: (ON C A)
    <(0 SUCCEEDED
    >ASSERTING 07: (ON C TAB)
    <07 SUCCEEDED
    >ASSERTING 08: (A TOP)
    <08 SUCCEEDED
  <MOVE SUCCEEDED: INMOVAL
<01 SUCCEEDED: ((ON C TAB))
>GOAL 09: (ON B C)
  >(THCONSE MOVE: (ON B C)
    >GOAL 010: (ON B (ANY X))
    <010 SUCCEEDED: ((ON B TAB))
    >GOAL 011: (C TOP)
    <011 SUCCEEDED: ((C TOP))
    >GOAL 012: (B TOP)
    <012 SUCCEEDED: ((B TOP))
    >PART 013: B IS PUSHED ON C
    >(ERASING 010: (ON B TAB)
    <(10 SUCCEEDED
    >(ERASING 09: (C TOP)
    <(09 SUCCEEDED
    >ASSERTING 014: (ON B C)
    <014 SUCCEEDED
  <MOVE SUCCEEDED: INMOVAL
<09 SUCCEEDED: ((ON B C))
>GOAL 015: (ON A B)
  >(THCONSE MOVE: (ON A B)
    >GOAL 016: (ON A (ANY X))
    <016 SUCCEEDED: ((ON A TAB))
    >GOAL 019: (B TOP)
    <019 SUCCEEDED: ((B TOP))
    >GOAL 020: (A TOP)
    <020 SUCCEEDED: ((A TOP))
    >PART 021: A IS PUSHED ON B
    >(ERASING 016: (ON A TAB)
    <(22 SUCCEEDED
    >(ERASING 020: (B TOP)
    <(23 SUCCEEDED
    >ASSERTING 024: (ON A B)
    <024 SUCCEEDED
  <MOVE SUCCEEDED: INMOVAL
<015 SUCCEEDED: ((ON A B))
>VAL: ((ON A B))
>
>:
:STOP
```

Gli autori ringraziano in particolare Robert Kowalski, che li ha indirizzati in questo campo di ricerca e li ha aiutati a chiarire il ruolo della logica formale, Marco Somalvico e gli altri ricercatori dello MP-AI Project, che li hanno aiutati nella comprensione dei linguaggi orientati ad obiettivi.

Manoscritto pervenuto il 18 dicembre 1974.

BIBLIOGRAFIA

[1] P. J. Hayes: *Robotologic*. In: Machine Intelligence, vol. 3, Meltzer and Michie Eds., Edinburgh University Press, Edinburgh, 1970.

[2] N. J. Nilsson: *Problem Solving Methods in Artificial Intelligence*. McGraw-Hill Book Company, New York, 1971.

[3] C. Green: *Application of Theorem Proving to Problem Solving*. Proc. of First I.J.C.A.I., Washington, 1969.

[4] R. E. Fikes, N. J. Nilsson: *STRIPS: a new approach to the Application of Theorem Proving to Problem Solving*. « Artificial Intelligence », vol. 2, n. 3/4, 1972.

[5] T. Winograd: *Procedures as a Representation for Data in a Computer Program for Understanding Natural Language*. MAC TR 84, M.I.T., Cambridge, Massachusetts, 1971.

[6] S. E. Fahlman: *A Planning System for Robot Construction Tasks*. « Artificial Intelligence », vol. 5, 1974.

[7] C. Hewitt: *Description and Theoretical Analysis (using schemata) of PLANNER, a Language for Proving Theorems and Manipulating Models in a Robot*. MAC TR 258, M.I.T., Cambridge, Massachusetts, 1972.

[8] G. J. Sussman, T. Winograd, E. Charniak: *MICROPLANNER Reference Manual*. Project MAC, AI Memo 203, M.I.T., Cambridge, Massachusetts, 1970.

[9] R. E. Fikes, P. E. Hart, N. J. Nilsson: *Learning and executing generalized robot plans*. « Artificial Intelligence », vol. 3, 1972.

[10] A. Tate: *Interacting goals in problem solving*. AISB European Newsletter, Issue 18, 1974.

[11] G. J. Sussman: *The virtuous nature of bugs*. AISB Summer Conference, Brighton, 1974.

[12] R. Kowalski: *Logic for Problem Solving*. Dept. of Computational Logic, Memo 75, University of Edinburgh, 1974.

[13] G. Gini, M. Gini: *Programming a robot by goal-oriented languages*. WOGSC 3th International Congress of Cybernetics and Systems, Bucharest, Romania, 1975.

[14] L. Siklossy, J. Dreussi: *An Efficient Robot Planner which generates its own Procedures*. « Proc. of 3th I.J.C.A.I. », Stanford, 1973.

[15] D. Warren: *WARPLAN: a system for generating plans*. Dept. of Computational Logic, Memo 76, University of Edinburgh, 1974.

[16] G. Battani, H. Meloni: *Interpreteur du Language de Programmation PROLOG*. U.E.R. de Lumigny, Université d'Aix-Marseille, 1973.

[17] E. Sacerdoti: *Planning in a hierarchy of abstraction spaces*. « Artificial Intelligence », vol. 5, 1974.

[18] R. Kowalski: *Predicate logic as a programming language*. IFIP Congress 74.

[19] D. V. McDermott, G. J. Sussman: *The CONNIVER Reference Manual*. Project MAC, AI Memo 259a, M.I.T., Cambridge, Massachusetts, 1972.

[20] G. J. Sussman, D. V. McDermott: *From PLANNER to CONNIVER - A genetic approach*, Fall Joint Computer Conference, 1972.

[21] J. F. Rulifson, J. A. Derksen, R. J. Waldinger: *QA4, a Procedural Calculus for Intuitive Reasoning*. S.R.I., AI Tech. Note 73, S.R.I. Menlo Park, California, 1972.

[22] D. Bobrow, B. Raphael: *New programming languages for Artificial Intelligence*. « Computing Surveys », vol. 6, 3, 1974.

[23] G. Gini, M. Gini, M. Somalvico: *Emergency recovery in Intelligent Robots*. 5th International Symposium on Industrial Robots. Chicago, Illinois, 1975.