Consiglio Nazionale delle Ricerche
ISTITUTO DI ELABORAZIONE DELLA INFORMAZIONE
PISA

Pubblicazione n. A 73-16

---

G. LEVI - A. MARTELLI - U. MONTANARI - C. MONTANGERO - G. PACINI - G. PRINI
F. SIROVICH - F. TURINI

# PROGETTAZIONE ED IMPLEMENTAZIONE DEL LINGUAGGIO LISPP PER INTELLIGENZA ARTIFICIALE

---

Estratto da: Atti del Convegno UMI su « Rapporti tra ricerca matematica pura e ricerca matematica applicata » (Siena, settembre, 1973).

London, 1972.

4 Proceedings of ACM Conference on proving assertions about programs, Las Cruces, New Mexico, Jean 6-7 1972.

5 Aiello M., ed al., Demantic concepts in Problem Solving, Planner Languages and Question Answering, Monografia dell'I.E.I. Pisa 1973.

6 Bobrow D.G., Requirements for Advanced Programming Systems for List Processing, Comm. ACM, Vol. 15, N. 7, July 1972 pp.618-627

7 Eljas, B., ed Al. An assessment of Techniques for Proving Program Correctness, ACM Computing Surveys, Vol.4, N.2, June 1972 pp. 97-147.

8 Montanari U., Epistemologia dichiarativa o procedurale: il problema della rappresentazione in A.I. Atti del conv. di Informatica Teorica dell'A.I.C.A. pp. 81-106, Editrice Tecnico Scientifica, Pisa, Marzo 1973.

9 Hewitt C., Description and Theoretical analysis (using schemata) of Planner: a language for proving theorems and manipulating model in a Robot. M.I.T., A.I. Technical Report, aprile 1972.

10 Sussman G.J., Winograd T., Micro-Planner Reference Manual, M.I.T. A.J., Memo 203 A. 1972.

11 Sussman G.J., Mc Dermott O.J., From Planner to Conniver: a genetic approch Proc. FJCC, pp. 1171-1179, 1972.

12 Rulifson J.F., Derksen J.A., Waldinger R.J., QA4: A procedural calculus for intuitive reasoning Tecnical Note, SRI project 8721, Stanford Research Institute, 1973.

13 Devies, D.J.M., Popler 1.5., A Revised Planner-like system in Pop-2: Unpublished Notes, School A.I., Theoretical Psychology Unit, Edimburgh, 1971.

14 Asirelli P., Montangero C., Pacini G., Implementazione del nucleo di un sistema LISP orientato verso la elaborazione di modelli semantici. Nota Tecnica C72-5 I.E.I., Pisa , 1972.

15 Montangero C., Pacini G., LISPP an interactive extended system. Atti del S.E.A.S. Spring Technical Meeting 1973 on interactive computing. Rimini 1973.

16 Turini F., Un modello di linguaggio per la programmazione non deterministica. Tesi in Scienze della Informazione, Pisa, febbraio 1973.

Un attore definisce una classe di espressioni; i pattern possono contenere attori, e il confronto di un pattern con una espressione ha successo se le sotto-espressioni che corrispondono ad un attore appartengono alla classe da esso definita.

Sia la ricerca di espressioni che la chiamata di funzioni per pattern-matching sono intrinsecamente cause di non determinismo: è chiaro che più di una espressione o di una funzione può soddisfare un pattern. Anche sotto questo aspetto il sistema quindi richiede una struttura del controllo più complesso di quella di un normale sistema LISP.

Gran parte delle estensioni apportate al sistema LISP sono relative al controllo; il sistema non mette a disposizione regimi di controllo rigidamente definiti, bensì un piccolo numero di primitive che consentono la semplice programmazione di regimi complessi come coroutine, processi paralleli e backtracking. Ad esempio una delle primitive permette di salvare lo stato del sistema ad un generico stadio del calcolo, e quindi di ripristinarlo, permettendo tra l'altro, di programmare facilmente il backtracking, regime in cui il fallimento di una scelta impone di riprendere la elaborazione dallo stadio precedente la scelta. La programmabilità del regime di controllo, consente l'interazione tra tentativi successivi (nel caso di backtracking) o paralleli.

Il progetto della organizzazione della memoria e del pattern matching è in fase di ultimazione. La parte riguardante il controllo è stata completamente progettata [14, 15, 16] ed è in avanzata fase di realizzazione.

Bibliografia.

1 McCarthy, J., ed al. LISP 1.5. Programmer's Manual, the M.I.T. Press, Cambridge Mass. 1963.

2 Weissman C., LISP 1.5. Primer, Dickenson Publishing Co. Belmont Calif., 1967.

3 Proceedings of 2nd Internat. Conf. on Artificial Intelligence,



# PROGETTAZIONE ED IMPLEMENTAZIONE DEL LINGUAGGIO LISPP PER INTELLIGENZA ARTIFICIALE.

G. Levi - A. Martelli - U. Montanari - C.Montangero - G. Pacini - G. Prini - F. Sirovich - F. Turini.

## 1. Introduzione

Scopo di questa memoria è di illustrare il progetto di un sistema per la manipolazione di contesti semantici, attualmente in corso di realizzazione.

La necessità di elaborare modelli semantici (cioè strutture complesse di dati simbolici rappresentanti relazioni concettuali) sorge in connessione con alcuni problemi di applicazioni di intelligenza artificiale e di programmazione quali:

- i sistemi di domanda e risposta;
- la comprensione dei linguaggi naturali;
- la dimostrazione automatica di teoremi;
- la costruzione di piani di azione per robots;
- la dimostrazione manuale, interattiva ed automatica della correttezza, terminazione, equivalenza etc. di programmi e schemi di programmi;
- la sintesi di programmi;
- lo sviluppo di linguaggi di programmazione in cui sia possibile esprimere il contesto semantico in cui il programma si colloca.

Per una trattazione di questi argomenti si rimanda ad atti di recenti congressi [3, 4] ed ad alcuni lavori di rassegna [5, 8].

Le ricerche svolte in questo campo hanno messo in evidenza la necessità di sviluppare nuovi linguaggi con particolare aspetti sia descrittivi (per esprimere i modelli) sia procedurali (per guidare i processi deduttivi). Tali linguaggi vengono detti "Linguaggi per Theorem provers" o linguaggi "tipo PLANNER" dal primo sistema di questo tipo sviluppato recentemente al MIT [9, 13] . Tra le possibilità più interessanti offerte da tali linguaggi, ricordiamo:

---

Istituto di Elaborazione della Informazione - C.N.R. - Pisa

- la possibilità di memorizzare in forma dichiarativa (assiomatica) asserzioni e loro proprietà, e di ritrovarle per "pattern matching": mediante un pattern, cioè una espressione non completamente specificata, è possibile ritrovare una qualunque, o tutte, le asserzioni la cui forma è compatibile con il pattern;
- la possibilità di ottenere una procedura partendo da una o più asserzioni. Il programma ottenuto può essere visto come un aspetto procedurale delle asserzioni, o come una regola di inferenza ottenuta "compilando" gli assiomi corrispondenti; ad ogni procedura può essere associato un pattern: per tentare di risolvere un problema, rappresentato da una espressione, possono essere attivate, per pattern matching, tutte le procedure il cui pattern è compatibile con l'espressione; in altri termini tutte quelle adatte allo scopo;
- la possibilità di lasciare non specificate alcune scelte, ottenendo così programmi non deterministici. Durante l'esecuzione il programma sceglierà una sola alternativa (a caso, o mediante una euristica programmata) ma dovrà essere in grado di ritornare sui propri passi (backtracking) e di effettuare un'altra scelta in caso di fallimento;
- la possibilità di creare più processi contemporaneamente attivi. Ciò permette di eseguire programmi per scopi distinti in modo concettualmente indipendente (cioè senza forzare una sequenzializzazione non insita nel problema) e, d'altra parte, di effettuare vari tentativi in parallelo per raggiungere uno stesso scopo (breadth first search). La suddivisione delle risorse tra i processi (ad esempio: tempo di calcolo) viene effettuata da un programma supervisore, tenendo conto di una euristica adattata al problema.

E' stato deciso di realizzare il sistema come una estensione del LISP [1, 2], soprattutto per la semplicità concettuale, la



flessibilità e la capacità di questo linguaggio di rappresentare programma e dati mediante la stessa struttura a lista. La natura delle estensioni è tale da consigliare l'implementazione di un nuovo interprete LISP, piuttosto che la modifica di un interprete esistente. Il nuovo interprete, LISPP [13, 14] è stato scritto in un linguaggio ad alto livello.

Nel seguito verranno descritte le caratteristiche principali del progetto.

## 2. Organizzazione della memoria, pattern matching e tecniche di controllo.

Un contesto semantico è un insieme di espressioni, a cui possono essere associate proprietà. Le espressioni devono quindi essere rappresentate internamente in modo analogo agli atomi LISP: ad ogni espressione corrisponde un unico nome interno che individua la lista delle proprietà ad esso associate. Per ogni riferimento ad una espressione è necessario risalire dalla rappresentazione esterna al nome interno, cioè alla posizione di memoria della lista delle proprietà; due tecniche convenienti sono la codifica mediante tabelle "hash" e l'uso di una rete di discriminazione; la seconda tecnica risulta anche particolarmente adatta per la ricerca delle espressioni mediante pattern matching.

Il livello più semplice a cui il pattern matching può essere realizzato è quello in cui il pattern è una espressione contenente simboli di variabile; questo è il caso di sistemi come QA4, Popler, MicroPlanner e Conniver: l'algoritmo di matching coincide con l'algoritmo di unificazione del calcolo dei predicati del primo ordine. Nel sistema in sviluppo, seguendo alcune proposte del Planner, si intende adottare una tecnica di pattern-matching più sofisticata mediante l'introduzione di attori.