:- gulp!

BOLLETTINO

Gruppo ricercatori e utenti di logic programming

Numero 2 1987

Indice

I materiali per la pubblicazione devono essere inviati alla redazione presso:

ILC - CNR
via della Faggiola, 32
56100 Pisa

GULP, un anno dopo
Messaggio del Presidente

Gli ultimi tre anni hanno visto lo sviluppo di un rilevante interesse verso la programmazione logica, che fino a poco prima era solo oggetto di culto di una sparuta pattuglia di ricercatori. Tale interesse è dimostrato dall'enorme successo delle ultime conferenze europee e americane, che hanno visto una straordinaria partecipazione industriale.
In Italia voglio segnalare le dimensioni assolutamente non previste dell'adesione al GULP, che ha avuto tra i soci fondatori quasi tutte le imprese che contano nell'informatica avanzata. Si tratta di sola curiosità, di moda passeggera o di semplice sottoprodotto del grande interesse verso le tecnologie in qualche modo collegate all'intelligenza artificiale? Dietro al fenomeno c'è un pò di tutto questo, ma anche qualcosa di più. Due sono a mio parere le ragioni del successo:

- le potenzialità di questa tecnologia, che fornisce risposte (anche se parziali) a necessità presenti in vari campi applicativi;
- il fatto che l'idea è stata venduta sul mercato, quando già si poteva appoggiare su prodotti industriali ben ingegnerizzati e realmente utilizzabili "in produzione".

(*dalla prima pagina*)

Sul primo aspetto, bisogna notare che la programmazione logica non è solo uno dei tanti paradigmi utilizzabili nell'ingegneria della conoscenza. I linguaggi logici, infatti, pur permettendo di definire regole del tipo IF...THEN..., hanno molte caratteristiche in comune con i linguaggi tradizionali o altre tecnologie, come le basi di dati. Fra le caratteristiche principali:

- il sistema di inferenza è definito da un vero e proprio interprete efficiente. Il "programma" può essere compilato, ottenendo delle prestazioni ancora più elevate.
- esistono delle tecniche interessanti (metaprogrammazione) per affrontare in modo pulito i problemi di rappresentazione della conoscenza, di risoluzione di problemi e di definizione degli strumenti di supporto.
- il linguaggio si interfaccia "naturalmente" con i DB relazionali, estendendone le funzionalità (viste, data base deduttivi, vincoli, "un vero linguaggio di programmazione" per le interrogazioni), e rendendoli direttamente utilizzabili in nuove applicazioni "intelligenti".
- i programmi logici puri possono essere "letti" in modo puramente dichiarativo (ignorando del tutto gli aspetti algoritmico-operazionali). Tale lettura si basa su una semplice e naturale semantica logica ed è la base di alcuni importanti usi dei linguaggi logici: tra questi, l'uso come linguaggio di specifica dei requisiti (prototipazione rapida) e quello come linguaggio "di programmazione" per l'utente finale non esperto (o per la formazione).
- i linguaggi logici sono considerati interessanti come linguaggi di programmazione altamente paralleli e ciò è importante nella prospettiva dello sviluppo di nuove architetture o sistemi della V Generazione.

Anche dal punto di vista della ricerca teorica, la programmazione logica offre un contesto formalmente molto pulito, nel quale quasi tutti gli approcci classici dell'informatica si possono confrontare su problematiche che hanno un immediato interesse applicativo.

Sul piano degli strumenti disponibili, programmazione logica è sinonimo di PROLOG, il linguaggio che si è ormai decisamente affermato, almeno al di fuori degli U.S.A., come il principale linguaggio per la programmazione per regole. PROLOG non è solo un linguaggio per rappresentare la conoscenza attraverso regole, ma è un vero e proprio linguaggio di programmazione, con cui si possono manipolare strutture dati complesse.

Esistono oggi ormai molte versioni industriali di PROLOG: pur non esistendo uno standard, le versioni più comuni differiscono per particolari secondari, anche se sono ormai molto diverse dall'implementazione originale sviluppata a Marsiglia nei primi anni settanta. Fra le implementazioni più popolari, tutte distribuite in Italia e, quasi tutte, già utilizzate in progetti industriali:

- le varie versioni del Prolog di Edinburgo, l'ultima delle quali (NIP-Prolog) è molto efficiente e dotata di un compilatore incrementale,
- quelle di micro-PROLOG, sviluppate ed industrializzate soprattutto su personal computers (particolarmente ricche per applicazioni tipo sistemi esperti),
- MPROLOG, che fornisce una struttura di moduli ed un ragionevole ambiente di sviluppo, ed è disponibile in numerosi ambienti operativi,
- Quintus-Prolog, compilato sulla macchina virtuale di Warren, che sembra destinata a diventare

un target standard anche per altri compilatori,

- BIM-Prolog, dotato di compilatore e di interfaccia verso un paio di DBMS relazionali e sviluppato da una società belga in un progetto ESPRIT.

Come ho già accennato, le imprese italiane sono in questo momento molto attive in questo settore, anche se con obiettivi ed accentuazioni diversi. È significativo, a questo proposito, quanto emerso dalla giornata organizzata dal GULP sulle attività italiane in programmazione logica nell'ambito dei progetti europei di ESPRIT, su cui si riferisce in questo stesso Bollettino. Varie aziende (CSELT, Elsag, Enidata, Olivetti, Sipe, Systems & Management) sono già impegnate, talora in posizioni di leader, su ambiziosi progetti, che, tuttavia, sembrano ancora interessare solo indirettamente i rispettivi mercati. È naturalmente un passo necessario, prima di passare ad introdurre le nuove tecnologie nei prodotti applicativi. Tutti coloro che credono nelle potenzialità di questa tecnologia si augurano che il prossimo passo si compia al più presto.

In questo quadro, cosa ha fatto e cosa può fare il GULP? Le iniziative non sono state molte, ma, indirettamente, credo che il GULP abbia contribuito ad un maggiore inserimento nella comunità internazionale della ricerca italiana (sia universitaria, che industriale) nel settore. Nell'ultimo anno si sono tenuti in Italia vari Workshops o Conferenze, dedicati a temi di programmazione logica, che hanno permesso una ampia partecipazione italiana. Le informazioni relative non sono state purtroppo fatte circolare attraverso i canali del GULP, ma hanno comunque raggiunto un numero notevole di soci. Fa eccezione l'unica iniziativa ufficiale del GULP sui progetti ESPRIT.

Approfitto dell'occasione per informare che si terranno prossimamente la 4th International Conference on Logic Programming (Melbourne, fine maggio) e il 4th Symposium on Logic Programming (San Francisco, fine agosto). È stata da pochi mesi costituita la Association for Logic Programming (il GULP internazionale!), cui faranno capo d'ora in poi la International Conference on Logic Programming ed il Journal of Logic Programming. Discuteremo la possibilità di fare del GULP anche la sezione italiana di tale Associazione. È comunque importante che la presenza italiana nella Associazione sia significativa fino da ora. Invito pertanto (a titolo personale) i ricercatori aderenti al GULP ad iscriversi (una copia del modulo di iscrizione e le informazioni relative compaiono in questo Bollettino).

Per quanto riguarda i soci più interessati alle aree applicative, il GULP ha effettivamente fatto finora molto poco, pur riconoscendo la difficoltà di iniziative promozionali in questa direzione. Una iniziativa interessante potrebbe essere una mostra dedicata a dimostrazioni di strumenti e di prodotti (o prototipi) applicativi. Anche di questo discuteremo nelle sedi ufficiali del GULP.

Vogliate gradire i miei più cordiali saluti e arrivederci a Torino. *G. Levi.*

UTILIZZO DEL PROLOG IN ATTIVITA' DI INTELLIGENZA ARTIFICIALE IN ELSAG

Maurizio Del Canto e Francesca Fusconi
ELSAG
Via Puccini 2 - 16154 Genova

L'intelligenza artificiale e' uno dei filoni principali su cui e' attivo il Servizio Ricerca Centralizzata dell'Elsag. In particolare un numeroso gruppo di ricerca e' da diversi anni impegnato nel campo dei Sistemi Basati sulla Conoscenza.

L'esigenza di utilizzare un linguaggio di programmazione che permettesse un rapido sviluppo di sistemi sperimentali e di prototipi in quest'area di ricerca ha portato in modo naturale alla scelta del Prolog, e piu' precisamente dell' M-Prolog della SzKI di Budapest in ambiente VAX/VMS.

Il primo sistema per cui il Prolog e' stato utilizzato e' EVA, sistema Esperto di Visione Artificiale (si veda "Programmazione Logica nel progetto EVA" di Fusconi, Oneto, Viano, atti GULP '86). EVA e' stato realizzato con lo scopo di indagare quale puo' essere l'apporto di tecniche di I.A. alla soluzione di problemi nel campo della visione atificiale.

EVA e' un sistema di riconoscimento di scene contenenti pezzi meccanici variamente disposti, finalizzato ad un possibile utilizzo in ambiente industriale. Formalismo di rappresentazione e meccanismi di interpretazione sono i problemi centrali di una tale applicazione.

La facilita' di trattare dati simbolici ha reso possibile basare le descrizioni delle immagini su primitive simboliche, e in questo modo ottenere le caratteristiche volute di sinteticita' e localita' della rappresentazione, come pure una relativa generalita' e univocita' del linguaggio.

Inoltre i meccanismi di unificazione e controllo propri del Prolog hanno consentito una immediata traduzione della strategia di gestione del sistema e una conseguente flessibilita' e facilita' di aggiornamento dei criteri decisionali, sfruttando la quale e' possibile adattare i criteri stessi alle molteplici soluzioni che si possono presentare.

Questo lascia intravedere la possibilita' di ulteriori evoluzioni della versione attualmente implementata, quali l'introduzione di un trattamento esplicito dell'incertazza nel valutare i risultati dei vari passi di analisi per poi indirizzare le successive eleborazioni o l'ampliamento delle fonti di informazione prese in esame. Una interessante prospettiva che si sta esaminando e' la parallelizzazione degli algoritmi, particolarmente adeguata a questo tipo di applicazione rivolta all'analisi di dati facilmente scomponibili in sottoinsiemi indipendenti (zone dell'immagine, singole regioni, singoli contorni, ecc.).

Un'altra interessante applicazione del Prolog e' stata fatta nell'area di ricerca riguardante i sistemi basati su regole. E' stato infatti implementato in questo linguaggio un prototipo di generatore di sistemi esperti che sperimenta metodi di rappresentazione della conoscenza e metodi di inferenza adatti alla progettazione e configurazione assistita di sistemi flessibili di lavorazione.

Il sistema realizzato, in particolare, permette di definire sia conoscenza dichiarativa che procedurale mediante due distinti tipi di regole.

Le regole dichiarative specificano relazioni tra fatti per permettere di dedurre fatti da altri fatti e sono utilizzate in backward-chaining per verificare la consistenza di una ipotesi (un fatto la cui struttura e' parzialmente specificata).

Le regole procedurali necessitano invece di una interpretazione in forward-chaining che permette di utilizzare le regole quando non si conosce la struttura dell'obiettivo.

Entrambi i tipi di regole e i fatti hanno potuto trovare nel Prolog una rappresentazione interna molto vicina a quella "Italian-like" con cui sono inserite dall'utente, con evidente vantaggio per la chiarezza e lo sviluppo dei programmi.

La possibilita' di rappresentare concetti e sottostrutture di concetti per mezzo di variabili simboliche e di definire nuovi simboli durante la fase operativa hanno contribuito al raggiungimento

di una buona capacita' espressiva. Molto utile e' inoltre risultata l'integrazione nel tool di strategie di interpretazione delle regole di tipo diverso, il che conferisce al sistema la capacita' di adattarsi facilmente a problemi di vario tipo.

Per quanto riguarda l'implementazione di tali strategie, l'utilizzo del Prolog e' risultato particolarmente vantaggioso per il backward-chaining, poiche' in questo caso le strategie proprie del linguaggio possono essere adattate in modo semplice e naturale.

Maggiori problemi si sono presentati nella riproduzione di strategie forward, che implicano un utilizzo poco naturale delle caratteristiche del linguaggio. Questo produce da un lato una parziale perdita della chiarezza espressiva e dall'altro una maggiore efficenza deduttiva.

Un altro problema di non immediata soluzione e' stata l'implementazione dei quantificatori, in quanto essi non sono trattati esplicitamente dal Prolog.

La scelta del Prolog ha comunque consentito una prototipazione rapida ed una soddisfacente sperimentazione delle strategie.

Il Prolog e' stato anche scelto come linguaggio di implementazione nell'ambito del progetto ESPRIT P865: Non Monotonic Reasoning Techniques for Industrial Planning Applications, a cui l'ELSAG pertecipa insieme a BATTELLE, AERITALIA e ITALCAD.

In tale progetto e' stato previsto di utilizzare metodi diversi di trattamento delle conoscenze (interpretazione di regole secondo strategie forward e backward, propagazione di vincoli, ragionamento non-monotono, ecc) al fine di creare un ricco ambiente contenente strumenti per la pianificazione, con particolare attenzione alla pianificazione in ambiente industriale e piu' precisamente alla pianificazione dei processi di produzione, con una interfaccia diretta a sistemi CAD.

Una buona parte dei temi di ricerca che si dovranno affrontare nella realizzazione del progetto, che e' attualmente al primo dei quattro anni previsti, e' in corso di studio; tra questi, tecniche di ragionamento per default e sistemi di truth maintenance, ragionamento qualitativo e temporale, logica sfumata.

Ci si aspetta dal Prolog un valido supporto per la sperimentazione rapida delle tecniche indagate, ed anche un efficace ambiente di supporto per le applicazioni finali, rivolte ad utilizzatori inesperti di tecniche di programmazione, per i quali e' indispensabile un approccio quanto piu' gradevole ed immediato al sistema.

Da subito, comunque, si e' evidenziata come caratteristica positiva del Prolog l'elevata modularita' e conseguente possibilita' di sviluppo incrementale dei programmi. Questo e' particolarmente apprezzabile in un progetto internazionale, in cui i compiti implementativi sono ripartiti tra partners fisicamente lontani e risolve, per lo meno nella fase di studio delle metodologie, problemi di coordinamento altrimenti non trascurabili.

Infine il Prolog e' utilizzato in una ricerca attualmente in corso riguardante le tecniche di ragionamento in presenza di incertezza.

Un primo passo per l'impostazione della ricerca e' stato lo studio delle soluzioni offerte dai primi sistemi esperti che consentivano il ragionamento con conseguenze incerte, quali MYCIN, INTERNIST, PROSPECTOR.

Sono stati chiariti i vantaggi e gli svantaggi delle varie soluzioni adottate, sia per quanto concerne l'introduzione nella rappresentazione della conoscenza del fattore di incertezza, sia per quanto riguarda i meccanismi inferenziali utilizzati.

In particolare, la teoria delle funzioni di combinazione ha permesso di unificare gran parte dei metodi che trattano l'incertezza, ed e' stato realizzato in Prolog uno shell il cui motore inferenziale realizza questi metodi e li rende disponibili per applicazioni.

Una prima applicazione di tale sistema e' stata nel campo del riconoscimento di immagini, e piu' precisamente per la localizzazione del blocco indirizzo di un oggetto postale.

Le varie tecniche di ragionamento approssimato messe a disposizione dal sistema, quali il procedimento bayesiano, la metodologia basata su funzioni di combinazione (usata in MYCIN) e le funzioni di membership delle logiche fuzzy, sono tutte risultate utili nelle varie fasi del problema.

Quindi il linguaggio Prolog risulta oggi uno dei linguaggi piu' utilizzati per lo sviluppo delle ricerche piu' avanzate di Intelligenza Artificiale in ELSAG. Gli aspetti di chiarezza espressiva e prototipazione veloce sono quelli che maggiormente lo fanno preferire. Problemi di efficenza e di disponibilita' in ambienti target rendono invece al momento meno appetibile il suo utilizzo in programmi di tipo piu' applicativo.

SIMULATION STUDY OF A MULTIPROCESSOR PROLOG ARCHITECTURE

I.FUTÓ, C.PECEBOIS, I.DURAND, C.SIMON, B.BONHOURE

Laboratoire "Languages et Systemes Informatiques"
Université Paul Sabatier
118 Route de Narbonne - 31062 Toulouse Cedex - France

ABSTRACT

This paper summarizes the first results obtained by simulating the COALA (Calculateur Orienté Acteurs pour la Logique et ses Applications) machine:Actor-oriented Computer for Logic and its Applications.

During the simulation process, two basic software systems were used. The first one was an emulator of COALA written in Pascal and executed on an SM-90 computer and the second one was a system simulation written in T-PROLOG, a simulation language based on PROLOG.

The emulator executes program statements sequentially and its use permits to get the execution times for the elementary operations on an MC68000 microprocessor. The measures obtained during emulation were used for the simulation of the basic operations in parallel. By the use of the T-PROLOG simulator, it was possible to compare the impact of different numbers of processors, supposed to be MC 68000, and different network topologies on the overall performance of COALA.

1. INTRODUCTION

Simulation is the only method which can supply information about the possible working of a planned, not yet existing, object. During the simulation process, we do experiments with a model of the planned object and we try to extrapolate the obtained results.

After defining the basic concepts of COALA, we wanted to validate the different algorithms of its components and to have some ideas about its future behaviour when executing PROLOG programs.

The first step was the realization of an emulator in Pascal on the SM-90 computer. This emulator executes the PROLOG programs using the algorithms of COALA in a monoprocessor environment. These algorithms are executed sequentially. No impact of the interprocessor communications and network topologies are taken into account during emulation. Once the algorithms and the basic concepts of COALA were validated by this emulator, the next step was to try to take into account the effects of different topology types.

In order to achieve this goal, we needed a simulator of a higher logical level: the system level. We choose the T-PROLOG languge [FUT 84] to create such a simulator for the following reasons:

- compactness of programs;
- availability;
- experience in writing T-PROLOG programs;
- need for unification (already existing in T-PROLOG);
- short deadlines.

T-PROLOG supports the execution of processes in parallel by means of an internal scheduler and it uses an internal clock to measure times and durations.

The first version of the simulator, including:

- a precompiler;
- an interpreter;
- a network generator;
- an output generator

was achieved within three months. The present paper describes the simulator and the first results

produced by its use.

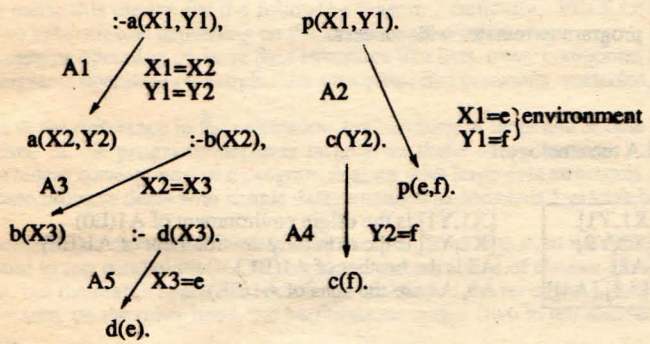
2. COALA: AN ACTOR-ORIENTED COMPUTER FOR LOGIC AND ITS APPLICATIONS

COALA uses the connection graph representation of PROLOG programs as its internal model. The arcs are distributed among several processors and the resolution is executed by means of:

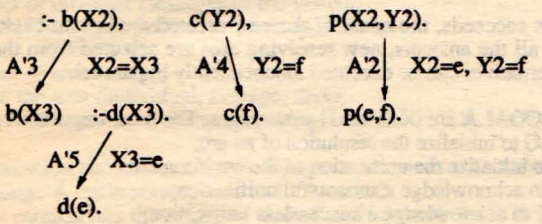
- messages;
- deletion of arcs;
- creation of new arcs.

In the following a simple PROLOG program is executed using the connection graph model.

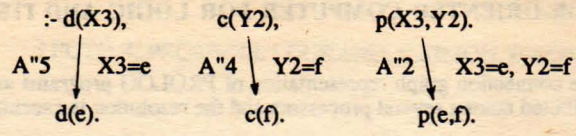
Initial graph:



First resolution step, elimination of A1:



Second resolution step, elimination of A'3:



Third resolution step, elimination of A"5:

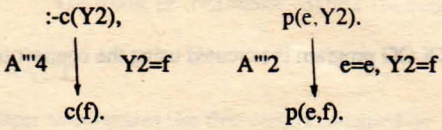
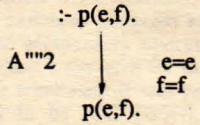


Figure 1: Execution of the connection graph (continues)

Fourth resolution step, elimination of A"'4:



Fifth resolution step, elimination of A""2:

[] program terminates with success.

In COALA terminology:
A1

E0 = [X1,Y1]	[X1,Y1] is the origin environment of A1(E0)
EE = [X2,Y2]	[X2,Y2] is the extremity environment of A1(EE)
BL = [A2]	A2 is the brother of A1(BL)
SL = [[A3],[A4]]	A3, A4 are the sons of A1(SL)

At each moment of the execution of a PROLOG program, the arcs under elimination (resolution) must know their brothers and their sons to send them the environments for unification.

If the unification succeeds, the name of the new created arc is sent back to the resolving one. After the arrival of all the answers, new resolving arcs are selected from the list of recently created new arcs. OR-alternatives can be executed independently in parallel.

The activities in COALA are initialized by messages. The most important messages are:
- RESOLVE-REQ to initialize the resolution of an arc;
- UNIFY-REQ to initialize the unification of the environments;
- UNIFY-ACK to acknowledge a successful unification;
- UNIFY-NACK to acknowledge a successless unification;
- BROTHERS-REQ for garbage collection;
- DESTROY-REQ to delete arcs corresponding to successless alternatives.

A Processor in COALA is composed by two components (Processing Units):
- Unification Unit (UU);
- Resolution Unit (RU).

The Unification Unit handles the messages:

- UNIFY-REQ;
- BROTHERS-REQ;
- DESTROY-REQ;

while the Resolution Unit handles the messages:

- RESOLVE-REQ;
- UNIFY-ACK;
- UNIFY-NACK.

For more detail about the structure and algorithms of COALA, see the companion paper [PER 86].

3. THE SIMULATOR OF COALA

The emulator consists of two independent parts:
- precompiler;
- interpreter.

Both parts are written in Pascal and can be executed on an SM-90 or a VAX 11/780 computer. The precompiler transforms a source PROLOG program into its equivalent connection graph and distributes the graph among the processors.

The emulator executes the resulting connection graph. The use of the emulator allows to get statistics about the static behaviour of the system.

For experimentation and comparison purposes, we chose two programs:
- quicksort;
- pc database.

In Appendices A and B the corresponding PROLOG programs are given in detail.

We made this choice for the following reason. Practically, PROLOG programs can be divided in two basic classes depending on the structures they operate on:
- programs operating on large data structures like lists, trees, compound terms;
- programs operating on simple data structures, like constants, variables, numbers.

Due to the difference in the unification and handling of large and simple data structures, the execution time of the programs depends largely on these data structures. In our choice, the quicksort problem corresponds to a program dealing with large data structures (list of lists), while the pc database problem deals with simple data structures as constants and variables.

Another basic difference between the two programs is that, in the quicksort one, which is used in general to test parallel PROLOG machines, the partitions of clauses are small (at most three alternatives), but the search space becomes large because of the recursive use of clauses. In the pc database problem, on the other hand, the partitions are larger (two to ten alternatives), but they are not recursive.

By the use of the emulator, we can collect statistics about the:
- number of different messages of the system;
- length of messages (individual, average, max);
- number of reductions;
- number of arcs created;
- size of the arcs (individual, average, max);
- execution times of the elementary operations (individual, average, max).

The execution time of an elementary operation means the number of processor cycles to handle the message. A cycle corresponds to 10^{-7} second.
Table 2 summarizes the results of use of the emulator, that is the results of the emulation.

4. THE SYSTEM SIMULATOR OF COALA

To have ideas about the dynamic behaviour of COALA, we built a system simulator in T-PROLOG, a simulation language based on PROLOG.

Static interpretations of program execution given by the emulator are used by the simulation model in two ways, giving two kinds of simulation methods:

- loosely coupled simulation method;
- strongly coupled simulation method.

4.1 The loosely coupled simulation method

The scheme of the loosely coupled simulation method is given in figure 1. If this method is used, once the statistical data are collected no more emulation is needed; the simulator, on a statistical base, can execute PROLOG programs.

As a first step, we wrote a loosely coupled simulator in T-PROLOG. The simulator consistes of four main modules, as described by figure 2.

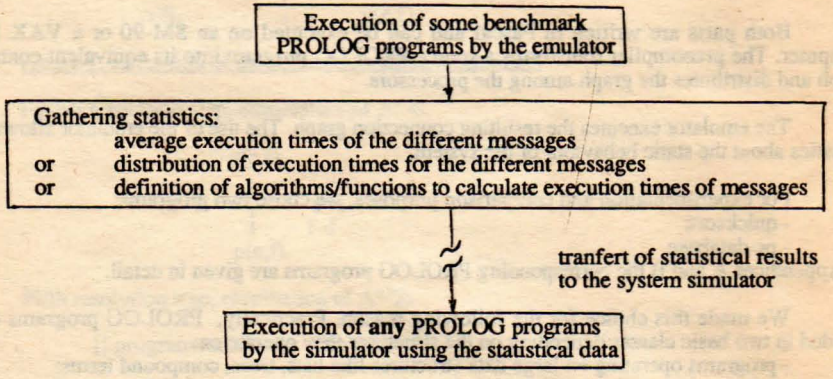


Figure 1: Loosely coupled simulation

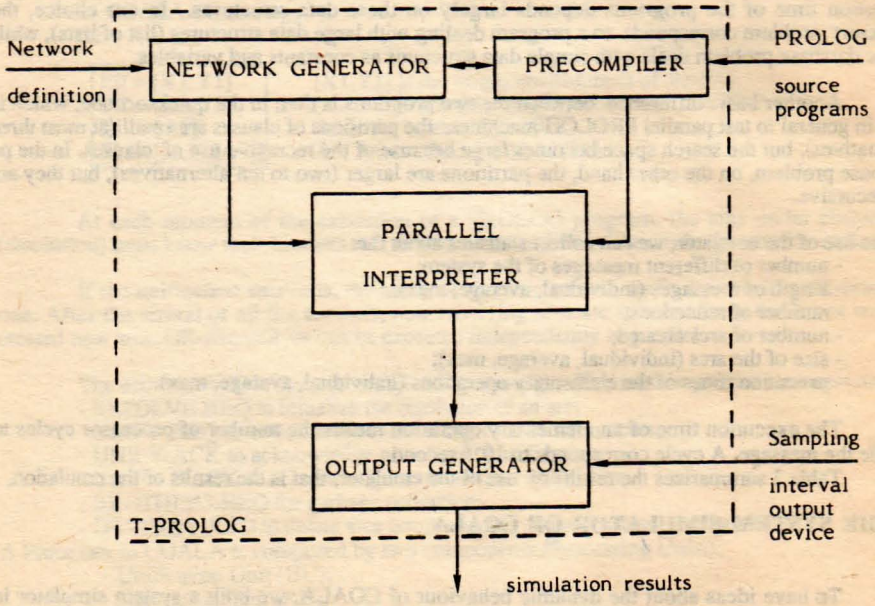


Figure 2: The loosely coupled simulator

The network generator accepts as input a 3D color graphical image of the network (processor space) with max 125 processing elements.

It can generate automatically:
- point to point;
- near-neighbor mesh;
- bus

topologies with a given number of processing elements.

The user can defin the type of channels:
- duplex;
- half duplex;
- transfer rate (in Mbit).

Obviously, the user can define any kind of networks in the 3D space by the use of the graphical input module.

In this Paper, the following topologies were used (for strongly coupled simulation):

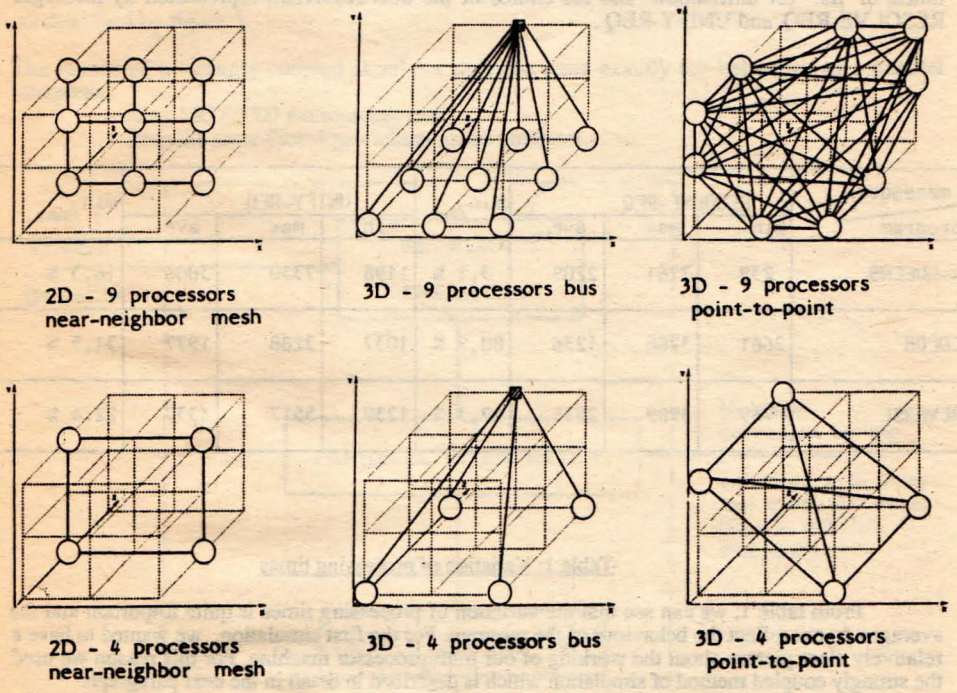


Figure 3: Network topologies for simulation.

The **precompiler** compiles a PROLOG source program into its equivalent connection graph and distributes the arcs among the processing units (defined by the network generator).

The **parallel interpreter** in T-PROLOG:

- assigns a process to each of the Unification and Resolution Units;
- assigns one/two processes to communication channels depending on their type (half-duplex / duplex);
- executes the program taking into account the
  - execution times of messages;
  - state of the channels (busy/free);
  - state of the processing elements (busy/free).

As T-PROLOG is a simulation language, it has built-in primitives to
- create processes: new (G,P,T1,T2);
- send messages: send (M);
- wait for a message: wait-for (M);
- suspend a process for a given time duration to simulate the elapsed time: hold (T).

These primitives are used for message passing and for simulating the execution times. An interval scheduler executes the processes in parallel in the simulation time of the system.

The **output generator** generates statistical tables about the execution of the PROLOG programs.

However, most of the existing simulations use the loosely coupled method [ONA 85]. We did not choose it because of the following considerations: in Table 1, we give some execution times, in µs, for unification and the choice of the next resolvent, represented by messages RESOLVE-REQ, and UNIFY-REQ .

messages / program	RESOLVE-REQ			min/max	UNIFY-REQ			min/max
	min	max	ave.		min	max	ave.	
4-QUEENS	239	7761	2209	3,1 %	1198	7330	3006	16,3 %
COLOR	2661	3288	1236	80,9 %	1037	3288	1977	31,5 %
REVERSE	769	3989	2535	19,3 %	1239	5517	3337	22,4 %

Table 1: Variation of processing times

From table 1, we can see that the variation of processing times is quite important and the averages do not reflect the behaviour of the program. For the first simulation, we wanted to have a relatively clear picture about the working of our multiprocessor machine. For this reason we used the strongly coupled method of simulation which is described in detail in the next paragraph.

4.2 The strongly coupled simulation method

The scheme of a strongly coupled simulation is given in figure 4.

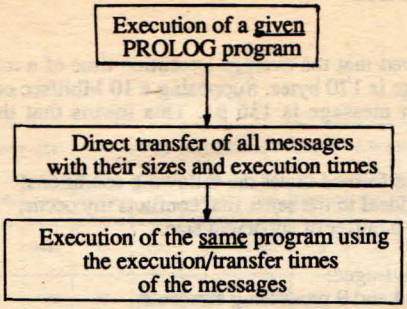


Figure 5: Strongly coupled simulation

The strongly coupled simulation consists of three modules:
- network generator;
- parallel interpreter;
- output generator

as described by figure 5.

The results of a strongly coupled simulator describe quite exactly the behaviour of a parallel interpreter:
- using MC 68000 microprocessors;
- using the same Pascal procedures as the emulator.

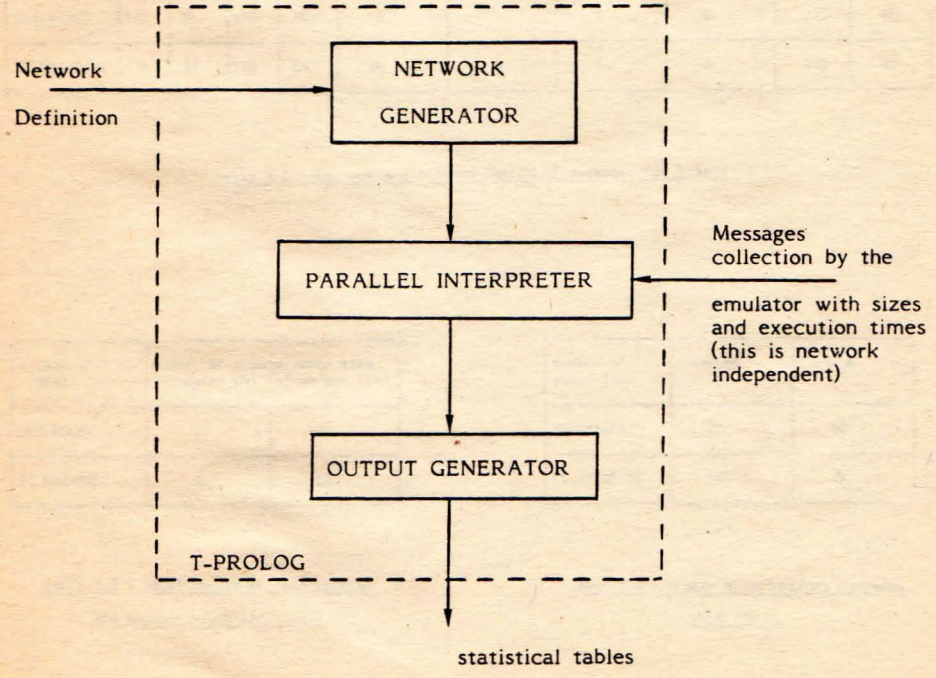


Figure 5: The strongly coupled simulator

5 - SIMULATION RESULTS

Measurements showed that the average execution time of a message is about 1000 mms. The longest size of a message is 170 bytes. Supposing a 10 Mbit/sec communication channel, the transfer time of the longest message is 136 μs. This means that the transfer time cannot be neglected.

Finally, the simulation was performed under the following conditions:
- the network is not ideal in the sense that conflicts my occur;
- each processor has a buffer of sufficient size.

As network topologies we envisaged:
- point to point with 4 and 9 processing elements;
- bus connection with 4 and 9 processing elements.

Simulation showed that no significant differences exist between bus and point to point connection in our case.

Tables 3.1 to 3.4 summarize dynamic results about the execution of quicksort and pc database programs with a bus topology.

STATIC RESULTS :

MESSAGES	UNIFY-REQ		UNIFY-ACK		UNIFY-NACK		RESOLVE-REQ		BROTHERS-REQ		DESTROY-REQ		TOTAL		
Times / Program	number	ave.	number	ave.	number	ave.	number	ave.	number	ave.	number	ave.	num-ber	time max	time ave.
QUICKSORT	91	3100	65	400	26	170	16	4200	34	32	64	620	296	7600	1400
PC DATABASE	174	1600	76	420	98	200	19	6000	46	36	66	310	479	11000	1000

Table 2.1 : Number and average execution times of messages (in µS)

Length of messages	UNIFY-REQ		RESOLVE-REQ		UNIFY-ACK	UNIFY-NACK	BROTHERS-REQ	DESTROY-REQ	TOTAL	
	max	ave.	max	ave.	max = ave.	max = ave.	max = ave.	max = ave.	max	ave.
QUICKSORT	173	93	159	70	10	8	7	6	173	40
PC DATABASE	135	81	116	84	10	8	7	6	135	40

Table 2.2 : Maximum and average length of messages (in bytes)

number of arcs	number of arcs created (N)	average size of an arc (LA)
QUICKSORT	47	276
PC DATABASE	66	168

Table 2.3 : Total number of arcs created and average sizes (in bytes)

number of reductions	PROLOG	COALA
QUICKSORT	25	16
PC DATABASE	30	19

Table 2.4 : Number of reductions compared with PROLOG

DYNAMIC RESULTS :

PROCESSING UNITS	ACTIVE IN PARALLEL						BUSY IN % OF TOTAL PROCESSING TIME					
number of processors	1		4		9		1		4		9	
number of processing units	2		8		18		2		8		18	
	max	ave.	max	ave.	max	ave.	max	ave.	max	ave.	max	ave.
QUICKSORT	2	1,1	6	2,5	10	3,2	100	55	75	31	55	18
PC DATABASE	2	1,3	8	3,9	14	5,1	100	65	100	49	78	28

$$P_A = \frac{\sum_{i=1}^{nu} T_{wi}}{T}$$

P_A = average number of working processing units
T_{wi} = total working time of processing unit i
T = execution time of the program
nu = number of processing units

Table 3.1 : Dynamic statistics

BUSY IN %	UNIFICATION UNIT						RESOLUTION UNIT					
number of processors	1		4		9		1		4		9	
number of processing units	2		8		18		2		8		18	
	max	ave.	max	ave.	max	ave.	max	ave.	max	ave.	max	ave.
QUICKSORT	89	89	54	47	60	26	25	25	18	14	26	9
PC DATABASE	77	77	77	61	88	35	51	51	55	35	42	21

$$L_A = \frac{\sum_{i=1}^{np} T_{Bi}}{np \times T}$$

L_A = average load of a processing unit
T_{Bi} = total working time of a processing unit
T = total execution time of the program

$np = \frac{nu}{2}$, number of processors

Table 3.2 : Load of processing units

ARCS	Number max of arcs / processor
QUICKSORT	13
PC DATABASE	24

Table 3.3 : Max number of arcs per processor

QUEUES	max number of waiting messages
QUICKSORT	6
PC DATABASE	26

Table 3.4 : Max number of waiting messages (4 processors)

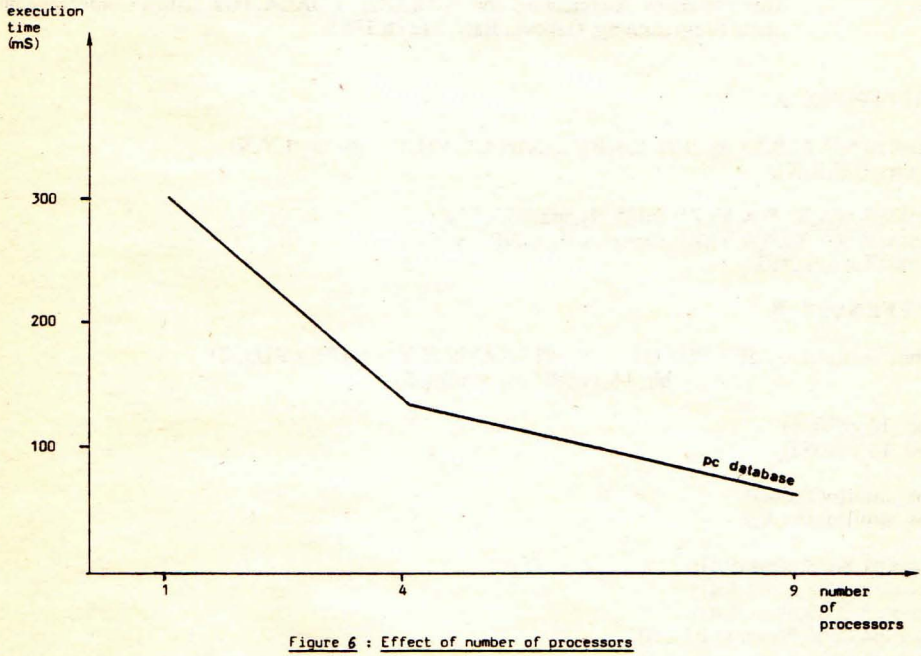


Figure 6 : Effect of number of processors

CONCLUSION

The results of dynamic simulation can be summarized as follows:

- execution times decrease favorably with increasing number of processors;

- the number of resolution steps is inferior or equal to that of a sequential interpreter.

However, a better distribution of tasks among the processing units is desirable. In the next future, we will investigate this aspect of our model and the effect of the different topologies when the execution times of processing units decrease.

REFERENCES

[FUT 84] I.Futo, J.Zeredi. System Simulation and Cooperative Problem Solving on a PROLOG basis. Implementations of PROLOG, J.Campbell ed., Ellis Hortwood Ltd, 1984, pp.323-364.

[ONA 85] R.Onai, M.Aso, H.Shimizu, K.Masuda, A.Matsumoto. Architacture of a Reduction-based Parallel Inference Machine: PIM-R. New Generation Computing 3, 1985, pp.197-228.

[PER 86] C.Pecebois, I.Futo, I.Durand, C.Simon, B.Bonhoure. An Actor Oriented Multiprocessor Architecture for PROLOG: COALA. First Italian Conference on Logic Programming, Genova, Italy, March 1986.

APPENDIX A

```
qsortx(*(H,T),S,X):-split(H,T,A,B), qsortx(A,S,*(H,Y)),qsortx(B,Y,X).
qsortx(nil,X,X).

split(H,*(A,X),*(A,Y),Z):-inf(A,H),split(H,X,Y,Z).
split(H,*(A,X),*(A,Y),Z):-supeg(A,H),split(H,X,Y,Z).
split(X,nil,nil,nil).
```

APPENDIX B

```
ibm_comp(PNAME,CPU,OS):-pnps(PN,PNAME,X,Y),pco(PN,CPU,OS),
                        bit_16_c(CPU),os_simil(OS).

bit_16_c(8086).
bit_16_c(8088).

os_simil(o(X,dos)).
os_simil(o(dos,X)).

pco(p1,8088,o(ms,dos)).
pco(p2,8088,o(ms,dos)).
pco(p3,8088,o(ms,dos)).
pco(p4,8086,*(o(dos_1,1),1)).
pco(p5,z80,cp_m).
pco(p5,8086,o(z,dos)).
pco(p6,8088,o(ms,dos)).
pco(p7,8088,cp_m_86).
pco(p7,z80,cp_m).
pco(p8,z80,cpm).
```

```
pnps(p1,agil,3000,s1).
pnps(p2,corona_p,2700,s2).
pnps(p3,corona-p,2500,s3).
pnps(p4,mbc_555,1100,s4).
pnps(p5,zenith_1,3000,s6).
pnps(p6,genie_16,2000,s6).
pnps(p7,rainbow,2500,s7).
pnps(p8,osborne,2000,s6).
```

PRESENTAZIONE DI PROGETTI ESPRIT A PARTECIPAZIONE ITALIANA RELATIVI ALLA PROGRAMMAZIONE LOGICA

a cura di S.Ghelfo, M.De Santis, F.Russo
ENIDATA, Bologna

IL 9 Giugno 1986 si e' tenuto a Milano, per iniziativa del G.U.L.P., un incontro tra i partner italiani di diversi progetti ESPRIT basati, almeno in parte, sulla programmazione logica. L'iniziativa di questa giornata di lavoro ha avuto origine dal proposito di coordinamento tra le diverse attivita' sulla programmzione logica attualmente sviluppate in Italia, che e' una della principali finalita' del G.U.L.P., e dalla constatazione che molte, se non la maggior parte, di queste attivita', orbitano intorno al progetto ESPRIT. E' sembrato quindi utile far convenire insieme rappresentanti di diversi progetti ESPRIT in modo da accendere il dibattito e favorire uno scambio di idee e di documenti fra persone che operano in contesti diversi.

L'incontro, organizzato e promosso dall'ENIDATA, si e' aperto con il saluto iniziale del presidente dell'ENIDATA, V. Gervaso, che ha sottolineato l'interesse della societa' verso le problematiche della programmazione logica; l'ENIDATA infatti partecipa a tre progetti ESPRIT centrati su questi temi.

Ha poi preso la parola Marco Valtorta, supervisore per conto della Comunita' Europea alla presentazione di nuovi progetti ESPRIT nell'area del Trattamento Avanzato dell'Informazione, che ha fornito alcuni dati generali sul progetto ESPRIT, e ha presentato una panoramica dei progetti attualmente in corso relativi alla programmazione logica e afferenti a diverse aree di interesse. Tra quelli senza partecipazione italiana si segnalano:

- LOKI (107,AIP) e ACORD (393, AIP), che usano Prolog come linguaggio per la rappresentazione della conoscenza; ACORD ha anche sviluppato una interfaccia tra il Prolog e una implementazione dello standard grafico GKS;
- PACT (951, ST), che prevede la costruzione di un semplice ambiente di programmazione Prolog basato sul PCTE;
- TOOL-USE (810, ST) e METEOR (432,ST), che studiano l'uso del Prolog come linguaggio per la produzione di prototipi;
- GIPE (348, ST) che affronta il problema della traduzione di specifiche algebriche in clausole, e viceversa;
- PROLOG III (1106,AIP), che costruisce un nuovo Prolog in cui diseguaglianze lineari sono risolte a livello dell'unificazione, e lo sperimenta nella costruzione di un sistema esperto.

Valtorta ha infine ribadito l'invito della Comunita' Europea alla presentazione di nuove proposte nel campo della programmazione logica.

Ha poi avuto inizio la presentazione di otto progetti ESPRIT a partecipazione italiana, che riassumiamo nel seguito in ordine di numerazione.

Progetto 26 - "Advanced Algorithms and Architectures for Signal Recognition and Understanding", presentato da G.Giandonato dello CSELT, cui partecipano CSELT (I), Thomson-CSF (F), GEC (UK), AEG (D), e l'Universita' di Torino (I) come subcontraente.

Obiettivo del progetto e' la realizzazione di un prototipo hardware / software per la sperimentazione di tecniche avanzate di elaborazione e comprensione di immagini e del linguaggio parlato. Il prototipo e' basato su di una architettura a due livelli: front-end acustico e riconoscitore lessicale; analisi sintattica, semantica, e comprensione. L'argomento affrontato comporta numerosi problemi, quali: computazioni con spazi di ricerca molto vasti; conoscenza rappresentata per mezzo di regole; elaborazione in presenza di incertezza; necessita' di euristiche e di metaragionamento; uso di backward e forward chaining; stategie di controllo flessibili e consistenti con l'uso del parallelismo; computazioni procedurali frammiste a computazioni inferenziali; elaborazioni in tempo reale. Sono quindi richieste capacita' molto elevate.

Il Prolog e' stato scelto per la programmazione del livello di analisi e comprensione del linguaggio perche' e' particolarmente adatto a rappresentare e trattare grammatiche, per le sue intrinseche capacita' inferenziali e per le potenzialita' di sfruttamento del parallelismo. Nel primo anno di attivita' del progetto e' stato condotto uno studio sui modelli di computazione parallela dei linguaggi logici, e sull'interazione tra i vari modelli e il problema affrontato, essenzialmente di parsing parallelo probabilistico, che condurra' alla scelta di un dialetto Prolog parallelo da eseguire su una macchina sperimentale multiprocessor in corso di realizzazione (la macchina e' di tipo MIMD, con memoria globale distribuita).

Per superare l'inadeguatezza del Prolog standard rispetto alle caratteristiche richieste dal problema vengono esaminate diverse possibilita': modifiche del linguaggio (ad es. backtracking intelligente); metaprogrammazione combinata con l'uso di tecniche di valutazione parziale e di compilazione in linguaggi logici concorrenti; traduzione delle regole in clausole in base al tipo di inferenza desiderato; estensioni per controllare il parallelismo OR.

Dopo il primo anno sono stati conseguiti diversi risultati: studio comprensivo su modelli paralleli di computazione per linguaggi di programmazione logica; metodi per tradurre alcune classi di algoritmi in dialetti Prolog paralleli; sviluppo di un compilatore in codice sul tipo della Macchina Astratta di Warren; definizione base del modello di computazione parallelo per il linguaggio target, consistente con l'architettura della macchina multiprocessor progettata.

Progetto 415 - "Parallel Architectures and Languages for Advanced Information Processing: a VLSI-directed Approach", presentato da G.Giandonato dello CSELT, cui partecipano: Philips (NL), AEG (D), BULL (F), GEC (DK), Nixdorf (D), CSELT (I), e i subcontraenti LIFIA (F), CWI (NL), Universita' di Pisa (I), Imperial College (UK), TUM (D).

L'obiettivo principale di questo progetto e' studiare e sviluppare architetture orientate al VLSI e con un alto grado di parallelismo per l'esecuzione di programmi nei linguaggi non convenzionali che si stanno imponendo (logici, funzionali, orientati ad oggetti). All'interno del progetto sono state individuate sei linee di ricerca fondamentali: macchine inferenziali parallele; macchine imperative parallele orientate a oggetti; macchine data-flow; macchine per data base deduttivi (alla Prolog); linguaggi funzionali di ordine superiore; integrazione tra logica e linguaggi funzionali. L'ultima di queste aree di ricerca e' l'argomento principale del sottoprogetto D, in cui sono coinvolti CSELT e Universita' di Pisa.

Per raggiungere l'integrazione tra programmazione logica e funzionale in modo operazionalmente semplice e semanticamente corretto sono stati studiati e approfonditi due approcci distinti, ma compatibili. Il primo si basa sull'uso di un formalismo in cui si fondono clausole Horn ed equazioni. In particolare e' stata formalmente dimostrata l'equivalenza del narrowing con la risoluzione di programmi flattened. Il secondo approccio puo' essere visto come una estensione del primo, in quanto tratta anche funzioni di ordine superiore. In questa direzione e' stato sviluppato il linguaggio applicativo deduttivo IDEAL, di cui e' gia' disponibile un primo prototipo, realizzato in C-Prolog, in cui le funzioni di ordine superiore sono tradotte in clausole per mezzo di un predicato apply, seguendo un'idea originale di Warren. Questo procedimento e' equivalente all'assiomatizzazione del I° ordine del $\neg$-calcolo e puo' essere molto efficiente se il Prolog sottostante e' in grado di indicizzare le clausole rispetto al primo argomento della testa. Il linguaggio finale sara' un'integrazione dei due approcci.

Progetto 419 - "Human Movement Understanding", presentato da C.Ruggero del D.I.S.T. di Genova, cui partecipano DIST (I), CAPTEC (IRL), Trinity College Dublin (IRL), University of Nijmegen (NL), Video Display System (I).

L'approccio all'analisi del movimento adottato nel progetto e' del tipo "static is basic", ed utilizza una ricostruzione delle superfici che compongono gli oggetti, completando e raggruppando le regioni individuate in entita' piu' complete. Il punto cruciale di questo processo sta nell'incompletezza dei dati a disposizione, derivante da ambiguita' e occlusioni che portano ad una molteplicita' di interpretazioni possibili.

E' stato quindi sviluppato un sistema, chiamato ESPOSE, implementato in C-Prolog su VAX-11/750, che esegue l'analisi delle occlusioni, ricostruendo la forma delle superfici visibili in base a regole percettive, senza richiedere conoscenza sugli oggetti esaminati.

Un modulo iniziale per l'elaborazione di immagini generalizzate produce immagini segmentate in regioni bidimendionali. Tali regioni sono poi usate come dati di ingresso primitivi e passati ad una teoria generale (indipendente dal dominio) della percezione visiva umana, codificata con regole Prolog. Il completamento delle regioni viene fatto applicando i principi di buona continuazione, di simmetria e di minima distanza ai lati delle regioni poligonali individuate.

Se il completamento proposto per una regione si dimostra consistente, viene

temporaneamente accettato e provoca l'asserzione di fatti nel database del Prolog; inoltre puo' attivare inferenze sulle regioni vicine, sfruttando quindi la propagazione contestuale dell'analisi delle occlusioni. Quindi la percezione di oggetti parzialmente occlusi e' vista come un processo di enunciazione e verifica di ipotesi.

Infine, le inferenze possono provocare conflitti; in tal caso viene usato il meccanismo di backtracking per rimuovere il conflitto, scegliendo la soluzione che non implica nessuna incosistenza locale e che conduce alla soluzione piu' semplice in termini di forma e numero degli oggetti coinvolti.

Progetto 432 METEOR - "An Integrated Formal Approach to Industrial Software Development", presentato da S.Ceri, relativamente all'attivita' svolta nell'ambito di questo progetto presso il Politecnico di Milano.

Obiettivo principale del progetto e' l'applicazione di metodi formali allo sviluppo di software per l'industria. Il lavoro svolto dal Politecnico di Milano in questo ambito si colloca nell'area di integrazione tra programmazione logica e basi di dati relazionali, ed esplora le potenzialita' di un approccio basato sull'algebra relazionale per la trasformazione dei programmi logici e l'ottimizzazione di interrogazioni rivolte in linguaggio logico ad un database deduttivo. Seguendo questo approccio, un programma logico viene automaticamente trasformato in un insieme di disequazioni (e quindi, usando la Closed World Assumption, in un insieme di equazioni) dell'algebra relazionale positiva. Analogamente, l'interrogazione viene trasformata in un'espressione relazionale. Sulle espressioni ottenute possono essere applicate tecniche standard di risoluzione e di ottimizzazione.

La tecnica di soluzione utilizzata si basa sul calcolo della chiusura di una espressione. I sistemi di equazioni possono essere risolti con metodi di diverso tipo: sostituzione, che puo' essere decisa sulla base del grafico delle dipendenze tra le relazioni; iterazione; calcolo della chiusura del prodotto cartesiano tra le relazioni.

I principali campi di ricerca in corso sono: ottimizzazione del calcolo della chiusura di una relazione; propagazione dei legami presenti nell'interrogazione, cioe' propagazione degli operatori di selezione all'interno della chiusura dell'espressione da valutare; tecniche di trasformazione dei programmi tradotti in forma algebrica (magic set, magic counting); modalita' di trattamento di funzioni.

Progetto 550 EPSILON - "Advanced Knowledge Base Management Systems Based on the Integration of Logic Programmming and Data Bases", presentato da C.Simonelli della LIST (Pisa), cui partecipano System & Management (I), Bense Computer System (D), CRISS (F), University of Dortmund (D), University C.Bernard (F), Universita' di Pisa (I), e i sottocontraenti Enidata (I) e Sipe Optimation (I).

L'obiettivo fondamentale del progetto e' la definizione di un ambiente per lo sviluppo di sistemi knowledge-based complessi, basati sull'integrazione di due tecnologie ormai consolidate: basi di dati relazionali e sistemi di programmazione logica (Prolog in particolare).

Altre due attivita' di ricerca svolte nel progetto riguardano la definizione del linguaggio di rappresentazione della conoscenza e la definizione di un insieme di strumenti per facilitare lo sviluppo e l'uso di KBMS. Infatti l'interprete di un linguaggio logico, come il Prolog, non e' adeguato, da solo, a definire sistemi knowledge-based. L'approccio al problema adottato in EPSILON consiste nella scelta di un unico meccanismo generale di rappresentazione della conoscenza, che permette amalgamazione di conoscenza e metaconoscenza, nonche' di una sola tecnica generale (metaprogrammazione) utilizzabile per la program- mazione sia di nuove regole di inferenza che di nuovi strumenti (realizzati con metaprogrammi).

Caratteristiche rilevanti di questo approccio sono la possibilita' di estendere il linguaggio, la macchina di inferenza e l'ambiente senza modificare l'interprete del linguaggio; la possibilita' di riutilizzare strumenti costosi (compilatori ottimizzanti); semplice realizzazione ed elevato grado di portabilita' dei metaprogrammi; possibilita' di adattare il linguaggio e la macchina di inferenza alle necessita' specifiche dell'utente.

Quindi la base di conoscenza in EPSILON e' vista come una collezione di teorie esistenti, in cui le relazioni tra teorie sono viste come meta-fatti, e le regole di inferenza vengono incorporate nei meta- programmi che definiscono le nuove operazioni.

I vantaggi della metaprogrammazione sono pero' in parte annullati dalla inefficienza introdotta. Sono percio' allo studio tecniche di valuta- zione parziale che mirano a compilare nel linguaggio base il linguaggio definito dai metainterpreti, e ad introdurre nei linguggi compilati le caratteristiche aggiunte dagli strumenti.

conoscenza sull'utilizzazione delle basi di dati, e per permettere all'utente di codificare direttamente la conoscenza in termini del dominio del problema.

L'ADE e' basato su di una stretta integrazione tra basi di dati relazionali e programmazione logica, che permette la realizzazione di interrogazioni logiche ricorsive e quindi di database di tipo deduttivo. Inoltre usa tecniche sofisticate di compilazione per ottimizzare il processo di interrogazione su database esistenti di notevole dimensione.

I risultati conseguiti durante il primo anno di attivita' nell'ambito di un precedente progetto 641) consistono nello sviluppo del formalismo OOPS per la rappresentazione della conoscenza; nella definizione di algoritmi efficienti per processare query logiche, che costituiscono un ambiente avanzato di basi di dati; nella creazione di una interfaccia grafica.

L'attivita' in corso e' incentrata su di un raffinamento del formalismo per la rappresentazione della conoscenza, basato su diversi studi di casi, e sua convalida attraverso la prototipizzazione; il perfezionamento dell'ambiente avanzato di basi di dati ADE; l'integrazione di KRF e ADE; l'aggiunta di una interfaccia intelligente.

GULP, un anno dopo
Messaggio della Segretaria

A un anno dall'inizio dell'attivita' del G.U.L.P., che possiamo far coincidere con il primo convegno sulla programmazione logica del marzo '86, viene naturale tentare un primo bilancio sullo sviluppo dell' associazione. Mi limitero' a un bilancio puramente da segretaria, fatto di cifre e indirizzi, lasciando ad altri (in particolare al presidente) il compito di fare un bilancio sull'attivita' svolta.

Durante il 1986 abbiamo registrato l'iscrizione di 166 soci, di cui 153 ordinari e 13 collettivi. Fra i soci ordinari, 12 sono studenti; gli altri sono quasi equamente suddivisi tra universita', istituti di formazione e istituti di ricerca da una parte, e piccole e grandi imprese dall'altra, con una lieve prevalenza del primo gruppo.

Per quanto riguarda la collocazione geografica, la distribuzione e' la seguente : Liguria 40; Toscana 34; Lazio 16; Piemonte 15; Lombardia 14; Veneto 11; Campania 8; Emilia 7; Sardegna 5; Calabria 4; Trentino 4; Marche 3; Umbria 3; Abruzzo 1; Sicilia 1. Da questi dati appare evidente una maggiore concentrazione in Liguria e nelle regioni adiacenti, certamente conseguenza del fatto che una buona parte dell'attenzione richiamata dall'associazione e' stata conseguenza del convegno svoltosi a Genova nel marzo '86. I dati relativi alla distribuzione delle iscrizioni nel tempo indicano pero' che l'interesse per il G.U.L.P. non e' stato suscitato unicamente dal convegno, poiche', anche dopo tale evento, l'afflusso di soci e' continuato, e ancora continua, piu' moderato ma costante, chiaro segno di interesse per gli scopi che l'associazione si prefigge e di fiducia nelle sue possibilita' di realizzarli.

Anche dal punto di vista finanziario, il bilancio dello scorso anno si puo' dire positivo. Infatti, nell'organizzare l'attivita' dell'associazione abbiamo sempre avuto cura di evitare le spese superflue, di limitare il piu' possibile quelle necessarie, di cercare finanziamenti ogni volta che è stato possibile, in modo che il G.U.L.P. non sia finanziariamente fine a se stesso, ma possa disporre di un minimo di risorse economiche che gli permettano di promuovere quelle attività di sviluppo e diffusione della ricerca (e applicazioni) nel campo della programmazione logica che sono previste dallo statuto.

Un anno positivo, dunque. Non rimane che augurarsi, anzi, augurarci, che dopo questo primo anno di rodaggio l'attivita' dell'associazione continui con sempre maggiore entusiasmo da parte di tutti.

Giuliana Dettori

RENDICONTO FINANZIARIO AL 31/12/86

ENTRATE		
quote sociali		
n.153 soci ordinari		4.590.000
n.13 soci collettivi		6.500.000
attivo del convegno		1.181.000
TOTALE ENTRATE	12.271.000	
USCITE		
spese notarili di registrazione		449.000
vidimazione registri		343.000
minute spese		66.150
anticipo convegno 1987		400.000
TOTALE USCITE	1.256.150	
SALDO ATTIVO	11.012.850	

ERRATA CORRIGE DELLA LISTA DEI SOCI G.U.L.P. PUBBLICATA SUL N. 1 DEL BOLLETTINO

Mauro Di Manzo
DIST
Via Opera Pia 11a
16145 Genova GE

Domenico Sacca'
CRAI
Via Bernini 5
87030 Rende CS

Giulio Cesare Giacobbe
Ist. di Filisofia
Via Balbi 4
16126 Genova GE

Letizia Tanca
Politecnico di Milano
Piazza L. da Vinci 32
20133 Milano MI

AGGIORNAMENTO AL 31.12.86 ALLA LISTA DEI SOCI G.U.L.P.

Paolo Barraco
Viale Marina 92
54038 Montignoso MS

Lucio Sansone
Dip. Informatica e sistem.
Via Claudio 21
80125 Napoli NA

Gianni Lazzari
IRST
c/o Fac. di Scienze
38050 Povo di Trento TN

Luigi Stringa
IRST
c/o Fac. di Scienze
38050 Povo di Trento TN

Paolo Mancarella
Dip. di Informatica
Corso Italia 40
56100 Pisa PI

Pietro Tumminello
Gruppo Inter. A.I.
Via Laurana 93
90143 Palermo PA

The Association for Logic Programming

October 1986.

Dear Colleague,

At this summer's Logic Programming Conference it was decided at a meeting of the current and new programming committees and the editorial board of the Logic Programming Journal to establish The Association for Logic Programming.

Keith Clark was elected President, and Robert Kowalski Treasurer, both for an interim period of two years.

The following people were proposed and accepted by the meeting as trustees of the Association:
Alain Colmerauer, Kazuhiro Fuchi, Herve Gallaire, Alan Robinson, Ehud Shapiro, Sten-Ake Tarnlund, David Warren.

It was also decided that the newsletter editor (currently Luis Pereira) and both the retiring chairperson and the next chairperson of the Logic Programming Conference would be officers of the Association for the period between conferences (currently these are Keith Clark, retiring chairman, and John Lloyd, new chairman).

The Articles of Constitution for the Association are at present being drafted.

Roles of the Society:

1. The Society will be the official sponsor of the International Conferences on Logic Programming, and will in future receive any profits from the conference and all profits and royalties from the sale of the proceedings of the conference. (We are discussing the possibility also of sponsoring the Symposium of Logic Programming currently sponsored by IEEE. In any case we will undertake to coordinate the two conferences.)

2. The Society will publish a Newsletter. (Initially edited by Luis Periera in Lisbon.)

3. It is intended that the Journal of Logic Programming will become the official publication of the Association.

Benefits for Members:

1. Members will receive the Newsletter free of charge.
2. Members will receive a 15% discount on the Logic Programming Conference fees.
3. Initially members will be offered a reduced rate subscription to the Journal; eventually this subscription may be included in an increased membership fee.

We very much hope that you will join the Association. Please complete the attached membership application form and return it to Robert Kowalski at Imperial College.

Yours sincerely,

Keith Clark.

The Association for Logic Programming

Membership Application Form

Name: _____

Address: _____

_____

_____

Tel. No. _____ **Net Mail:** _____

Affiliation: _____

Membership Categories:

Annual Subscription for year commencing Jan. 1987

___ Full member $25

___ Full time student $15

(copy of student id or letter from institution required)

Please enter my subscription to the Journal of Logic Programming

I enclose an additional $35 (USA and Canada) or ____

$43 (rest of the world) ____

Subscriptions can be paid in £ sterling. To compute the sterling subscription add 5% to the amount to be remitted and convert at the prevailing exchange rate.

Da inviare a G.Dettori, I.M.A - CNR, Via L.B. Alberti, 4, 16132 Genova

Scheda di Adesione al G.U.L.P.

Il Sottoscritto/a

Cognome .....Nome.....

Ditta/Ente/Università.....

Posizione ricoperta.....

Via.....n.....Tel.....

CAP.....Città.....Prov.....

chiede di essere iscritto al GULP - Gruppo Ricercatori e Utenti di Logic Programming,

in qualità di :

Socio Ordinario

Socio Collettivo, per la Ditta/Ente.....

Data.....Firma.....

Quota annuale di iscrizione:

Soci Ordinari.....Lire 30.000

Soci Collettivi.....Lire 500.000

Il versamento deve essere effettuato su c/c postale n. 11147568 intestato a:
Gulp - Pisa

Stampa: Giardini, Pisa.

Numero unico in attesa di registrazione.