:- gulp!

BOLLETTINO

gruppo ricercatori e utenti di logic programming

Primavera 1989

Indice

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

ILC - CNR
via della Faggiola, 32
56100 Pisa

Editoriale

Giunto al suo quarto anno di attività, il GULP aumenta le quote di iscrizione. Questo aumento non è dovuto al crescere dell'inflazione ma al fatto che si è presentata l'opportunità di offrire ai soci qualcosa di più, e precisamente l'iscrizione all'ALP associazione internazionale della programmazione logica. Questo significa che tutti i soci GULP, oltre agli usuali vantaggi offerti dalla nostra associazione (organizzazione di un convegno annuale, workshops, seminari, corsi) fruiranno dei vantaggi offerti dall'ALP (sconto sulla partecipazione alla conferenza internazionale e sull'abbonamento al Journal of Logic Programming, spedizione del bollettino). Questa nuova quota si rivela quindi molto vantaggiosa e ci è possibile offrirla solo grazie ad una convenzione tra GULP e ALP e al supporto, durante i tre anni passati, dei nostri soci collettivi il cui generoso contributo ci ha permesso, e ci permette, di coprire le spese eccedenti le quote di iscrizione dei soci.
Le quote di iscrizione ritoccate ci permettono di offrire anche un altro vantaggio. Su incarico dell'assemblea dei soci riunitasi a Roma in occasione del terzo convegno annuale, il comitato direttivo del GULP ha stipulato un accordo con la AI*IA (associazione italiana per l'intelligenza artificiale), in base al quale chi è iscritto ad

(continua)

entrambe le associazioni riceve uno sconto di lire 5000 su entrambe le quote di iscrizione, abbassando quindi la nostra a lire 35000.Anche la quota di iscrizione per gli studenti è stata riveduta, cambiando da lire 30000 per due anni a lire 20000 per un solo anno. In questo caso, purtroppo, non ci è possibile offrire lo sconto per i soci AI*IA, né l'associazione all'ALP, perché una quota così ristretta non ci permetterebbe neppure di coprire il versamento richiesto dall'ALP.

Questo numero del bolletino GULP, che purtroppo vi inviamo, insieme con il calendario 1989, con notevole ritardo, appare più ricco ed articolato dei numeri precedenti, grazie al contributo di diversi soci. Ne ringraziare quanti hanno collaborato, cogliamo l'occasione per invitare nuovamente tutti i soci ad inviarci contributi personali, sotto forma di recensioni, proposte di problemi scientifici ed organizzativi, informazioni su attività italiane e straniere connesse alla programmazione logica, lettere, desiderata, ecc.

La redazione

Stampa: Giardini, Pisa

Numero unico in attesa di registrazione

RELAZIONE SUL CONVEGNO GULP 88

di Maurizio Martelli

A Roma, nella grandiosa Aula Magna della Sapienza, si è svolto dal 9 al 13 Maggio scorso il Terzo Convegno Nazionale sulla Programmazione Logica (GULP '88) del GULP (Gruppo ricercatori ed Utenti di Logic Programming). Si tratta di un appuntamento annuale di questa associazione di recente costituzione che si è ormai conquistato uno spazio non indifferente nel panorama dell'informatica italiana.
Circa 200 partecipanti al Convegno ed agli ottimi Tutorials, una impeccabile organizzazione da parte del Dipartimento di Informatica e Sistemistica dell'Università di Roma "La Sapienza" e del CRAI sotto la attenta direzione di Daniele Nardi, un elevato livello scientifico dei contributi presentati e l'interesse dimostrato a questo convegno sia di settori universitari che industriali hanno fatto di questa occasione di incontro un evento decisamente interessante e positivo.

Una delle note più positive registrate durante questo convegno è l'alta partecipazione ed interesse di giovani ricercatori; ciò denota una tendenza di forte crescita di questo settore sia dal punto di vista teorico che di quello delle applicazioni.
Questo sviluppo è un oggettivo riconoscimento alla attività del GULP e del suo Presidente Prof. Giorgio Levi che hanno reso l'Italia una delle realtà più importanti a livello internazionale in questo settore.

Il GULP ha sempre inteso questi convegni come una vera occasione di incontro tra i vari gruppi che in Italia si occupano (in vari modi) di Programmazione Logica piuttosto che di copie in piccolo di grossi convegni internazionali; credo che questo spirito si respirasse a Roma (tranne qualche solitario intervento polemico decisamente stonato).
Il convegno voleva essere ed è stato anche una occasione per far fare esperienza ai ricercatori più giovani nel settore (a partire dal coordinatore e dagli insegnanti dei tutorials fino, quando era possibile, ai presentatori dei lavori).

Vi sono state tre relazioni invitate e 32 lavori presentati (su oltre 50 sottomessi); la provenienza geografica dei contributi è stata variegata (Amburgo, Ancona, Bologna, Cosenza, Genova, Milano, Napoli, Padova, Parma, Pisa, Roma, Torino, Udine) indice anche questo di una reale diffusione degli interessi in questo settore. Da notare anche che circa un terzo dei contributi veniva da progetti con diretta partecipazione industriale.

Dal punto di vista prettamente scientifico le tre giornate durante le quali sono stati presentati i contributi erano così organizzate: una prima giornata dedicata agli aspetti teorici e linguistici della Programmazione Logica, una seconda giornata dedicata alle applicazioni ed alle interazioni della Programmazione Logica con i Databases, ed una terza giornata dedicata alle interazioni della Programmazione Logica con l'Intelligenza Artificiale. Queste tre giornate sono state aperte rispettivamente da tre relazioni invitate di ottima qualità presentate da famosi ricercatori che ben hanno caratterizzato le succitate tematiche.

I ricercatori invitati sono stati:

Jean Louis Lassez (IBM, Yorktown), editor del Journal of Logic Programming e che ha dato importanti contributi sia alla teoria che agli sviluppi linguistici del settore, ha presentato un lavoro dal titolo *"From Unification to Constraints"* dove viene illustrata una delle più interessanti proposte di sviluppo della Programmazione Logica, il Constraint Logic Programming (CLP) Scheme. Questa proposta, ben sviluppata e coerente dal punto di vista teorico, è uno schema di linguaggi logici che ben si adatta a svariati settori applicativi. Il lavoro è una introduzione di questo approccio con alcuni esempi di instanziazioni dello schema.
Harve Gallaire (ECRC, Monaco), dirigente di uno dei più importanti centri di ricerca industriali in Europa e che ha dato importanti contributi nei settori di interazione tra Logic Programming, Databases e Object Oriented Programming, ha presentato un lavoro dal titolo *"Boosting Logic Programming"* dove vengono analizzate le varie proposte per migliorare la Programmazione Logica ma soprattutto per renderla realmente competitiva per grosse applicazioni industriali (integrazione con altri paradigmi linguistici, con Knowledge-bases, sviluppi di tecniche di controllo e di problem-solving più potenti, ecc.).
Raymond Reiter (University of Toronto), uno dei più noti ricercatori nel settore Database ed Intelligenza Artificiale, ha presentato un lavoro dal titolo *"On Integrity Constraints"* dove viene discussa e presentata una soluzione al cruciale problema degli Integrity Constraints nei databases e nei linguaggi per la rappresentazione della conoscenza. In particolare viene presentata una soluzione per gli integrity constraints statici nel linguaggio KFOPCE di Levesque.

Per quanto riguarda i lavori presentati, come già detto, sono stati mediamente di buona qualità ed avevano il pregio di descrivere progetti e ricerche in corso in modo da permettere un quadro abbastanza completo delle attività dei vari gruppi che in Italia si occupano di programmazione logica e settori collegati. Segue una telegrafica rassegna dei lavori così come si sono articolati nelle varie sessioni.

Teoria e Fondamenti

Sessione che ha presentato alcuni interessanti lavori teorici più o meno direttamente legati alla programmazione logica, ma che comunque affrontano temi che dovranno essere considerati in una accezione più ampia di questo settore. I legami tra un problema classico della IA e il LP ("Rilassamento in reti di vincoli e metaprogrammazione logica" un bel lavoro di F. Rossi e U. Montanari); il trattamento di conoscenza temporale ("Temporal Reasoning in a Hibryd System" di M. Poesio); alcuni problemi delle logiche non monotone ("Logiche non monotone su base intuizionista" di G. Fischer Servi); alcuni problemi semantici del calcolo dei predicati classico ("Strutture di simulazione come ambiente semantico per un calcolo classico" di A. Carbone); il Prolog come metalinguaggio ("Alcune osservazioni matematiche sulla metateoria del Prolog" di A. Vincenzi).

Linguaggi 1

Sessione che ha presentato in modo esauriente le tecniche all'avanguardia per migliorare le caratteristiche del LP e del suo ambiente di programmazione. Il problema della ottimizzazione dei programmi logici attraverso trasformazioni di programmi ("Un metodo per la specializzazione di programmi logici" di A. Bossi, N. Cocco e S. Dulli) ed analisi statica dei programmi usando tecniche di interpretazione astratta ("Optimization of Logic Programs Execution Based on their Static Analysis" di G. Codognet, M. M. Corsini e G. Filè); l'amalgamazione del livello oggetto con quello meta ("Alcune considerazioni sulla integrazione tra livello oggetto e meta nel linguaggio logico MI_Prolog" di L. Giordano, A. Martelli, I. Murgia e G.F. Rossi) e le tecniche di valutazione parziale ("Correttezza e completezza della valutazione parziale di programmi logici" di L. Fanti, G. Levi e S. Zanobetti).

Linguaggi 2

Sessione dedicata ai miglioramenti ed alle estensioni dei linguaggi logici oggi disponibili. I moduli ed i costrutti per la concorrenza ("Un' estensione dell'interprete PROLOG con modularità e concorrenza" di P. Mello, A. Natali, A. Rossolini e F. Russo); il parallellismo esplicito con estensione delle possibilità di controllo ("Un linguaggio OR-parallelo per una classe di applicazioni di IA" di G. Giandonato e G. Sofi); il trattamento delle eccezioni ("The concept of Exception Handling in VIP Prolog" di S. Dulli e E. Kuhn); l'utilità delle estensioni all'ordine superiore della programmazione logica ("Applicazioni di un linguaggio logico + funzionale di ordine superiore" di P.G. Bosco, C. Cecchi e C. Moiso).

Applicazioni 1

Sessione dedicata alle applicazioni orientate alla costruzione di sistemi esperti. Quelle basate su un interessante e nuovo approccio basato su architetture a blackboard ("Blackshell: una shell a blackboard in PROLOG" di A. Brogi, e M. Gaspari e "Un sistema esperto con architettura a blackboard per la fusione dati" di A. Brogi, R. Filippi, M. Gaspari e F. Turini); tecniche di valutazione parziale applicate a sitemi esperti ("Un valutatore parziale per l'ottimizzazione del sistema esperto ADES" di M. Cavalieri, R. Cremonini, E. Lamma e P. Mello); applicazioni del LP alla grafica per CAD ("Il PROLOG in ambiente di progettazione grafica" di V. Milanese); applicazioni per l'Ingegneria del Software ("L'uso della storia in un sistema per il controllo di accessi e versioni" di A. Huber Bachrich e D. Nardi).

Basi di dati

Sessione con contributi di frontiera tra LP, DB ed Object Oriented programming. Strategie di esecuzione backward e forward di programmi logici ("Metodi di riscrittura di clausole di Horn per la valutazione di predicati ricorsivi" di M.L. Sapino); integrazioni con linguaggi object oriented ("Un linguaggio orientato ad oggetti basato sulla programmazione logica" di S. Greco e P. Rullo); integrazione con i linguaggi algebrici ("LOAR: un sistema di interrogazione logico-algebrico" di M.A. Dispinzeri, L. Palopoli e C. Pizzuti); problemi specifici delle basi di dati deduttive ("Ottimizzazione della memoria per query non ricorsive" di G. Di Battista e A. Marchetti Spaccamela).

Applicazioni 2

Sessione dedicata ad uno dei più promettenti campi di applicazione della programmazione logica, quello della rappresentazione ed automazione delle leggi: "Programmazione in logica della legge sull'IVA" di R. Bertocchi, F. Frattini, G.A. Lanzarone e A. Maglia ed "Uso di linguaggi formali per la rappresentazione di testi normativi: il progetto PROLEG (PROlog applicato alle LEGgi)" di M. Andretta, M.G. Losano, M. Luganesi, N. Nannini e F. Zambon.

Intelligenza Artificiale

Sessione dedicata ad approcci od applicazioni di Intelligenza Artificiale che usano o possono essere di interesse per la Programmazione Logica. Il FOL, la metaconoscenza e la riflessione procedurale ("Bypasses fra contesti FOL" di F. Collovà, G. Criscuolo, G. Minicone, E. Minicozzi e A. Russo, "Una architettura riflessiva per i linguaggi logici" di S. Costantini e G.A. Lanzarone e "Formalizzazione di un caso di ragionamento qualitativo utilizzando insiemi di contesti" di L. Palomba); la fisica quantitativa ("Un approccio PROLOG alla Fisica naive: l'esempio dei processi qualitativi" di S. Bandini, M. Bruschi, M.G. Filippini e A. Molesini) ed i problemi della gestione dell'incertezza della conoscenza ("Metodi bayesiani per il trattamento

dell'incertezza in problemi di decisione: un'applicazione antropologica" di M. Chiogna, M. Gambera e A. Palareti).

Linguaggio Naturale

Un'ultima sessione dedicata ad un settore della IA che ha avuto una particolare rilevanza proprio per la nascita della programmazione logica come settore anche applicativo: il trattamento e l'analisi del linguaggio naturale. ("Un formalismo logico per la rappresentazione della semantica del linguaggio naturale" di P. Terenziani, L. Lesmo e P. Torasso, "Metodologie di rappresentazione e di controllo per un sistema di comprensione di testi" di F. Antonacci, M. Russo, P. Velardi e M.T. Pazienza ed "Interpretazione dell'Italiano in termini di Logic Programming" di G. Dondossola e G. Ferrari).

Advanced School on Foundations of Logic Programming

Alghero 19-23 Settembre 1988

Il GULP (Gruppo Utenti e ricercatori di Logic Programming) ha organizzato questa scuola avanzata di programmazione logica per dare una visione approfondita dei fondamenti di questo nuovo paradigma di linguaggio di programmazione sempre in costante evoluzione.

La diffusione ed il successo che, a partire dal progetto giapponese dei calcolatori della quinta generazione basati appunto sulla programmazione logica, questo settore ha avuto negli ultimi anni richiede, ad una associazione come la nostra, la diffusione delle basi e dei fondamenti di questo nuovo paradigma sia per gli aspetti più tradizionali che per le nuove espansioni che sta avendo.

Il convegno ha visto una partecipazione di circa 100 ricercatori italiani ed europei con qualche partecipazione anche da altri continenti. Del totale dei partecipanti circa un terzo era straniero e circa un terzo proveniva da realtà industriali.

I costi sono stati tenuti volutamente bassi per la caratteristica "non profit" della nostra organizzazione. Sono stati assegnati circa 25 contributi a giovani ricercatori per la partecipazione a costi ridotti (questo anche grazie al contributo del CNR).

Per quanto riguarda gli aspetti scientifici della scuola, riportiamo qui di seguito le relazioni sugli interventi dei docenti. Vogliamo comunque ringraziare i docenti che hanno accettato di partecipare alla scuola e che, oltre a dare un contributo di alta qualità, sono stati disponibili a svariati incontri e discussioni con i partecipanti.

Tutto, speriamo, é andato discretamente bene e di questo ringraziamo tutte le persone che ci hanno aiutato nella preparazione e gestione della scuola. Crediamo che questa iniziativa sia stata positiva sia come occasione per studiare i più importanti sviluppi del settore che come occasione di incontro tra svariati ricercatori, universitari ed industriali, che vi operano.

Roberto Barbuti e Maurizio Martelli

RELAZIONE SULL'INTERVENTO DI K.L. CLARK

di Antonio Brogi

K.L. Clark ha illustrato alcuni problemi relativi all'introduzione del parallelismo e della concorrenza nella programmazione logica. In particolare, anzichè trattare gli aspetti relativi al parallelismo implicito, ovvero a livello implementativo, sono stati descritti i problemi della concorrenza a livello dei linguaggi di programmazione.
Un linguaggio logico concorrente deve avere definiti dei meccanismi per trattare in modo esplicito il controllo del non determinismo e la sincronizzazione tra processi. Per fare questo è necessario introdurre dei costrutti extra-logici.
Nella programmazione logica pura, il tipo di non-determinismo è detto *don't know*, il sistema non sa come ottenere la soluzione di un problema e perciò ha bisogno di esaminare vari cammini alternativi. Nei sistemi concorrenti, spesso, si vuole trovare una sola soluzione; questo è detto non determinismo *don't care*, cioè il sistema non si preoccupa di tutte le soluzioni: ad ogni passo soltanto una alternativa (cioè una clausola) viene scelta e tutte le altre sono eliminate.

Le estensioni della programmazione logica orientate alla concorrenza che hanno riscosso maggior successo impongono dei vincoli sul meccanismo di unificazione, introducendo una esplicita distinzione tra i produttori ed i consumatori di una variabile logica.

Clark ha esaminato essenzialmente il linguaggio PARLOG (PARallel LOGic) di cui è autore insieme a S. Gregory.
Nel linguaggio PARLOG vengono imposti dei vincoli sul meccanismo dell'unificazione obbligando il programmatore a distinguere a livello del linguaggio quali sono i produttori ed i consumatori delle variabili, introducendo delle *dichiarazioni di modo*.

Per ciascun predicato (nome di processo) il programma contiene una dichiarazione che specifica, per ogni argomento, il modo di accesso, cioè se esso viene usato in ingresso o in uscita; non vi è la possibiltà di specificare un accesso bidirezionale. Comunque, soprattutto nelle ultime versioni del PARLOG, il modo di uscita ha perso il suo significato originario, che provocava il fallimento di un processo nel caso che esso trovasse le sue uscite già istanziate. Nella versione più recente, anche se un processo p è dichiarato essere un consumatore su x e x viene legata da un altro

processo, p non è forzato a fallire.

Un'altra caratteristica del linguaggio PARLOG è la possibilità di controllare il non determinismo tramite l'uso di condizioni, cioè imponendo che una alternativa possa essere scelta solo se certe condizioni date sono soddisfatte.

Il linguaggio è definito da clausole con guardie, ovvero clausole (quantificate universalmente) della forma:

$$H :- G_1,\ldots,G_m \mid B_1,\ldots,B_n$$

dove '|" è l'operatore di commit, H è la testa, $G_1,\ldots,G_m$ è la guardia e $B_1,\ldots,B_n$ è il corpo della clausola. La guardia rappresenta la condizione che deve essere soddisfatta affinché sia possibile usare la clausola per un passo di risoluzione.

Clark, dopo aver introdotto il linguaggio, ha mostrato numerosi esempi di programmazione soffermandosi sui problemi generali dell'espressività dei linguaggi logici concorrenti.

A questo proposito, sono stati fatti dei confronti con i linguaggi Concurrent Prolog e Guarded Horn Clauses (GHC), che rappresentano le due altre proposte più note in questo campo.

Tra i problemi aperti nel campo della programmazione logica concorrente, sono stati descritti quello della mancanza di una definizione di una semantica formale per i linguaggi logici concorrenti e quello dell'efficienza delle implementazioni.

RELAZIONE SULL'INTERVENTO DI J-L. LASSEZ

di Annalisa Bossi

Jean Louis Lassez ha tenuto una serie di lezioni su: "Logic Programming Language Scheme and Constraint Logic Programming". Il corso, durato complessivamente sei ore, è stato suddiviso in tre giornate.

La prima giornata è stata dedicata a richiami di semantica dei programmi logici. E' stato dato particolare risalto ai problemi inerenti la definizione ed il trattamento della negazione e alle relazioni esistenti tra le diverse nozioni di fallimento. Un articolo di base per i concetti introdotti in questa prima giornata è "Some issues and trends in the semantics of Logic Programs" in Proceedings of the 3rd Int. Conf. Logic Progr. London 1986, Springer Verlag LNCS 225.

Dimostrare che la programmazione con vincoli (Constraint Logic Programming) è la naturale evoluzione della programmazione logica è stato il filo conduttore della seconda giornata.

Dopo aver analizzato alcune estensioni di Prolog, Lassez ha dimostrato che le loro principali caratteristiche comuni possono essere catturate in uno schema di linguaggi di programmazione logica: CLP(X). Infatti ogni istanza di tale schema, ottenuta sostituendo il parametro X con una opportuna teoria dell'uguaglianza, fornisce un linguaggio logico di programmazione per il quale sono assicurate le proprietà semantiche caratterizzanti la programmazione logica:

- l'esistenza di un dominio canonico di interpretazione;
- l'esistenza di un modello minimo, di un minimo punto fisso per l'operatore T_P e l'uguaglianza delle corrispondenti semantiche;
- la consistenza e la completezza della risoluzione SLD per ciò che riguarda successi e fallimenti finiti;
- la consistenza e la completezza della regola di negazione per fallimento (NAF) relativamente ad una equa (fair) risoluzione SLD.

Procedendo nell'analisi delle principali proprietà dei programmi logici, Lassez ha sottolineato che queste non dipendono necessariamente dall'universo di Herbrand o dall'algoritmo di unificazione. Il ruolo di quest'ultimo può essere schematizzato nei due punti: (i) verifica della solubilità di un sistema di equazioni, (ii) costruzione di una rappresentazione esplicita delle soluzioni. Con questa ottica l'algoritmo di unificazione può essere assimilato ad un qualsiasi

algoritmo per la soluzione di un insieme di equazioni o, più in generale, di vincoli. L'assunto di Lassez è che la costruzione di una rappresentazione esplicita delle soluzioni non sia essenziale se una risposta positiva per la solubilità fornisce anche una rappresentazione implicita dell'insieme delle soluzioni. Ciò è assicurato su alcuni domini che egli ha chiamato "solution compact".
Ha quindi introdotto una nuova versione dello schema, uno SHELL di linguaggi di programmazione. Dato un dominio con le sue operazioni, ovvero una opportuna struttura algebrica, rimane individuato un linguaggio di programmazione specializzato a quel particolare dominio di applicazione. Un programma scritto in questo linguaggio è costituito da un insieme di regole con vincolo ovvero regole formate da una testa, un insieme di vincoli ed un corpo. L'insieme di vincoli costituisce una naturale definizione implicita di un insieme di oggetti del dominio (le soluzioni) e i predicati che compaiono nelle regole agiscono direttamente su questi ultimi. Ogni linguaggio dello SHELL gode automaticamente di tutte le proprietà semantiche proprie dei linguaggi logici convenzionali.

CLP(R) è un linguaggio di programmazione ottenuto dallo SHELL e specializzato all'aritmetica reale. Questo linguaggio è stato il tema principale dell'ultima giornata. Sono stati presentati alcuni esempi di programmi CLP(R) che hanno permesso di riconoscere i vantaggi della programmazione logica con vincoli rispetto a quella convenzionale. Tra questi vi è certamente la maggiore espressività. Problemi complessi possono essere rappresentati in modo naturale perchè il linguaggio permette di esprimere le proprietà direttamente nel dominio naturale del discorso anzichè richiedere di codificarle nelle liste del Lisp o nei termini del Prolog.

Lassez ha infine esposto alcuni problemi incontrati nella realizzazione di CLP(R) e derivati soprattutto dalla necessità di disporre di una implementazione efficiente. Tra questi vi è lo sviluppo di algoritmi per la soluzione incrementale di un sistema di vincoli e l'individuazione di rappresentazioni canoniche che, eliminando ridondanze, permettano di mantenere a livello minimale le dimensioni del sistema.

Il corso è stato seguito con grande interesse ed attenzione. I partecipanti hanno molto apprezzato sia il contenuto scientifico che la notevole simpatia e comunicatività del docente: iniziando il corso alle 8 del mattino, non è stata impresa da poco!

RELAZIONE SUGLI INTERVENTI DI J. LLOYD E L. STERLING

di Andrea Nicoli

La "Scuola avanzata di fondamenti di programmazione logica" ha presentato nel convegno tenutosi ad Alghero in Settembre due corsi alquanto complementari nel tema. L. Sterling ha tenuto interessantissime lezioni sulla costruzione di un Meta-Interprete per la programmazione logica e J. W. Lloyd ha illustrato i risultati raggiunti che forniscono una risposta teorica alle problematiche di consistenza e completezza poste dalla Meta-Programmazione. In questo articolo presentiamo una relazione introduttiva alle tematiche sviluppate nei due corsi. L. Sterling ha iniziato il corso introducendo elementi di terminologia e di stile di programmazione in Prolog che ci auguriamo si affermino presso gli utenti di programmazione logica. Ci ha ricordato che l'eleganza non e' un "optional", ma che e' un indice significativo della qualita' e leggibilita' del prodotto. Prima di affrontare il tema del corso ha introdotto una terminologia standard con i concetti di "modulation", "enhancement" e "mutation". Il concetto di "modulation" afferma che un programma Q e' un modulante di un programma P e vice-versa, se essi possono essere derivati l'uno dall'altro mediante una sequenza di "unfold" e "astrazione". Il concetto di "astrazione" corrisponde intuitivamente all'operazione di creare una nuova procedura Prolog per un codice gia' esistente. La semantica operazionale e dichiarativa di un programma P e del suo mutante Q e' preservata.

Esempio di "modulation".

```
programma P: son(X,Y) :- father(Y,X),male(X).
             son(X,Y) :- mother(Y,X),male(X).

programma Q: son(X,Y) :- parent(Y,X),male(X).
             parent(X,Y) :- father(X,Y).
             parent(X,Y) :- mother(X,Y).
```

Diciamo che un "enhancement" E di un programma P e' una alterazione della semantica operazionale di P che non modifica la sua semantica dichiarativa.

Esempio di "enhancement".

```
programma P:  is_tree(leaf(X)).
              is_tree(tree(L,R)) :- is_tree(L),
                                    is_tree(R).

programma E: sum_leaves(leaf(X),X).
             sum_leaves(tree(L,R),Sum) :- sum_leaves(L,Lsum),
                                          sum_leaves(R,Rsum),
                                          Sum is Lsum + Rsum.
```

Il concetto di "mutation", invece, implica una alterazione sia della semantica operazionale che della semantica dichiarativa.

Esempio di "mutation".

```
programma P:  is_tree(leaf(X)).
              is_tree(tree(L,R)) :- is_tree(L),
                                    is_tree(R).

programma Q: positive_tree(leaf(X)) :- X > 0.
             positive_tree(tree(L,R)) :- positive_tree(L),
                                         positive_tree(R).
```

Nella parte centrale del corso mostra come partendo dal semplice Meta_Interprete Vanilla sia possibile ottenere differenti componenti funzionali.

Meta_Interprete Vanilla.

solve(+Goal) <-

Goal is given the pure Prolog program defined by clause/2.

```
solve(true).
solve((A,B)) :- solve(A),
                solve(B).
solve(A) :- clause(A,B),
            solve(B).
```

Questo Meta-Interprete puo' essere letto sia da un punto di vista dichiarativo che procedurale. Letto in modo dichiarativo, afferma che la costante "true" e' vera. La congiunzione (A,B) e' vera se A e' vera e B e' vera. Un goal (A) e' vero se c'e' una clausola (A:-B.) nel programma tale che B e' vero. Letto in modo procedurale, afferma che il goal vuoto, rappresentato dall'atomo "true", e' sempre risolto. Nel caso della congiunzione afferma che per risolvere (A,B) deve risolvere (A) e risolvere (B). Per risolvere, in generale, un goal (A) deve trovare una clausola la cui testa unifica col goal e risolvere ricorsivamente il corpo della clausola. La lettura procedurale dimostra inoltre che viene mantenuta la scelta di selezionare nel corpo della clausola il goal piu' a sinistra e di mantenere il backtracking. Il nuovo Meta-Interprete e' sviluppato a partire dal Meta-Interprete Vanilla mediante una successione di "modulation", "enhancement" e "mutation". Si costruiscono piu' Meta-Interpreti che soddisfano le funzionalita' di costruire un albero di dimostrazione, la funzionalita' di contare la profondita' della computazione e la funzionalita' di ritornare il risultato della computazione. Le componenti funzionali cosi' costruite sono ora da ricomporre in un unico programma. L'algoritmo proposto da Sterling per la composizione e' guidato, quando possibile, da similarita' sintattiche. La sua operazione ha senso per clausole corrispondenti, cioe' per clausole che sono ottenute per "enhancement" della stessa clausola o come "enhancement" di clausole modulanti. Nella parte finale del corso introduce un esempio di costruzione di sistema esperto ottenuto componendo il Meta-Interprete con alcune componenti funzionali quali la facilitazione offerta dal potere interrogare l'utente quando il sistema non sa come comportarsi, quali la facilitazione offerta dal potere richiedere una spiegazione del ragionamento eseguito dal sistema. Come ultimo punto ha analizzato il problema dell'efficienza del Meta-Interprete. Come

soluzione ha proposto un algoritmo per la valutazione parziale del programma. Questa soluzione che si sta diffondendo tra gli studiosi di queste problematiche e' stata accuratamente analizzata nei suoi risvolti teorici da J. Lloyd. L'idea e' che, partendo da un programma P ed un goal G, una parziale valutazione genera un nuovo programma P' che e' specializzato per il goal G. La speranza e' di avere una risposta al goal G piu' efficiente per P' che per P, mantenendo la stessa risposta per P e P'. Possiamo dire, come illustrato nell'esempio successivo, che un programma P' parziale valutazione di un programma P rispetto al goal G si puo' ottenere rimpiazzando le clausole in P, la cui testa contiene uno dei simboli di predicato che compaiono nel goal G, con una parziale valutazione del goal G nel programma P.

```
programma P: grand_mother(X,Y) :- mother(X,Z),
                                  parent(Z,Y).
             grand_father(X,Y) :- father(X,Z),
                                  parent(Z,Y).
             parent(X,Y) :- mother(X,Y).
             parent(X,Y) :- father(X,Y).
             mother(sue,simon).
             mother(sue,patrick).
             mother(monica,sue).

goal G:      grand_mother(X,simon).

valutazione parziale:
             grand_mother(X,simon) :- mother(X,Z),parent(Z,simon).
                                      !
             grand_mother(sue,simon) :- parent(simon,simon).
                                        !
             grand_mother(sue,simon) :- father(simon,simon).

programma P':
             grand_mother(sue,simon) :- father(simon,simon).
```

Il nuovo programma P' ottenuto dalla valutazione parziale puo' essere usato con un incremento di efficenza per rispondere a goal del tipo (?:-grand_mother(X,simon).) . I programmi ottenuti per valutazione parziale, presentano dei comportamenti problematici, quando vengono utilizzati per soddisfare un goal diverso da quello coinvolto nella valutazione parziale. J. Lloyd ha presentato alcuni esempi in cui "completezza" e "soundness" non vengono preservate. La sua proposta per risolvere il problema e' di caratterizzare il goal da soddisfare rispetto al goal che ha originato la valutazione parziale. Egli introduce la seguente definizione: "Sia S un insieme di formule ed A un insieme finito di atomi. Si dice che S e' A-chiuso se ciascun atomo in S contenente un simbolo di predicato occorrente in un atomo A e' una istanza di un atomo in A." Dimostra che e' possibile preservare la correttezza nell'ipotesi che P'U{G} e' A-chiuso, dove P e' il programma normale, G il goal normale, P' e' la valutazione parziale di P rispetto al goal A. Con nostro sommo dispiacere, non si dispone ancora di nessun risultato positivo sulla completezza. Speriamo con queste pagine di soddisfare l'interesse al tema di quanti non hanno potuto partecipare al corso.

RELAZIONE SULL'INTERVENTO DI D. MILLER

di Laura Giordano e Gianfranco Rossi

Le lezioni di D.Miller alla scuola di Alghero, dal titolo "Logic Programming Based on Hereditary Harrop Formulas", hanno avuto come obiettivo principale quello di mostrare come sia possibile introdurre in un linguaggio di programmazione logica varie forme di astrazione quali le funzioni di ordine superiore, i tipi di dato astratti e i moduli, estendendo la teoria logica su cui si basa il linguaggio stesso. In particolare, Miller ha presentato un linguaggio logico, λProlog, che e' una estensione del Prolog in varie differenti direzioni: permette la programmazione higher-order, incorpora i λ-termini, include la nozione di tipo polimorfo, fornisce meccanismi per definire moduli e tipi di dato astratti. Queste caratteristiche del λProlog sono state ottenute sostituendo alla teoria classica del I ordine delle clausole di Horn (su cui si basa il Prolog), quella intuizionista delle higher-order hereditary Harrop formulas. Quest'ultima e' una estensione della logica delle clausole di Horn, che ne conserva molti degli aspetti computazionali.

La presentazione di Miller si e' articolata in sei lezioni, di cui la prima introduttiva e le altre incentrate rispettivamente sui seguenti argomenti: clausole di Horn di ordine superiore, unificazione di ordine superiore e generalizzazione della risoluzione SLD, Hereditary Harrop Formulas e prove "uniformi", Higher-Order Hereditary Harrop Formulas, moduli e scope lessicale.

Le estensioni della teoria logica che sono state prese in considerazione per definire il λProlog sono sostanzialmente di due tipi, quelle quantificazionali e quelle proposizionali. Le estensioni quantificazionali sono quelle relative all'introduzione della quantificazione sui simboli predicativi e funzionali, e quindi hanno a che fare con le clausole di Horn di ordine superiore e i λ-termini. Le estensioni proposizionali concernono invece l'introduzione di nuovi connettivi nei goal e nel body delle clausole e quindi hanno a che fare con le hereditary Harrop formulas.

La logica di ordine superiore, chiamata T, su cui si basa il linguaggio λProlog, e' formulata in un λ-calcolo tipato, la "Simple Theory of Types" di Church. Informalmente T ha le seguenti caratteristiche: tutte le costanti e le variabili hanno un tipo; e' permessa la quantificazione sui predicati e sulle funzioni; gli assiomi e le regole di inferenza per la versione classica (rispettivamente intuizionista) di T sono quelli della logica del I ordine classica (rispettivamente intuizionista) piu' la regola di inferenza della λ-conversione. La proof-theory di T ha una stretta rassomiglianza con quella della logica del I ordine; ad esempio esiste una generalizzazione del teorema di Herbrand che vale per T. Inoltre, un sottoinsieme di T, la logica delle clausole di Horn di ordine superiore, generalizza la logica delle clausole di Horn del I ordine, preservando varie delle sue proprieta' computazionali. In particolare, come nel caso del I ordine, e' possibile attribuire alle clausole una interpretazione procedurale.

La presenza dei λ-termini e delle regole di λ-conversione nel linguaggio, richiede che l'interprete del linguaggio sia capace di risolvere equazioni fra termini modulo la λ-conversione; questo rende necessaria l'unificazione di ordine superiore. Questa forma di unificazione e' notevolmente piu' complessa dell'unificazione nel I ordine: in generale non e' decidibile e, quando esiste un unificatore, puo' non esistere l'unificatore piu' generale. Tuttavia, la ricerca di un unificatore ha molte caratteristiche in comune con la ricerca di una prova di un goal da un insieme di clausole di Horn del I ordine ed e' possibile progettare un interprete per il linguaggio interallacciando le due ricerche.

Per quanto concerne le estensioni proposizionali che il λProlog introduce rispetto alle clausole di Horn, le piu' rilevanti consistono nel consentire che l'implicazione ed il quantificatore universale occorrano nei goal e nel body delle clausole. Si ottiene cosi' la classe delle Hereditary Harrop Formulas.

La logica delle Hereditary Harrop Formulas con la provabilita' intuizionista e' considerata adeguata come base dei linguaggi logici, come gia' la logica delle clausole di Horn con la provabilita' classica. In generale, data un teoria logica, essa viene considerata adeguata come base di un linguaggio per la programmazione logica se consente di vedere ogni connettivo logico come un operatore di ricerca in uno spazio di ricerca. Questo significa che la relazione di derivabilita' operazionale di un goal da un programma deve soddisfare alcune condizioni. Ad

esempio, se valgono le seguenti condizioni

AND Pl-G1$\wedge$G2 iff Pl-G1 and Pl-G2

OR Pl-G1$\vee$G2 iff Pl-G1 or Pl-G2 ,

dove P e' un programma e G1 e G2 sono goal, si possono interpretare i connettivi $\wedge$ e $\vee$ in un goal o nel body di una clausola del programma come nodi AND e OR in uno spazio di ricerca. La relazione fra i connettivi logici e le operazioni di ricerca puo' essere generalizzata agli altri connettivi. In particolare, affinche' l'implicazione, il quantificatore universale e quello esistenziale possano essere interpretati come operatori di ricerca, devono valere le seguenti condizioni:

AUGMENT Pl-D$\supset$G iff P$\cup$Dl-G

INSTANCE Pl-$\forall$xG iff Pl-[c/x]G, dove c e' una costante che non occorre in P o in G

GENERIC Pl-$\exists$xG iff Pl-[t/x]G per qualche termine chiuso t,

dove P e D sono programmi e G e' un goal. L'aggiunta di nuovi connettivi, in questo modo, corrisponde all'aggiunta di nuove primitive di ricerca al programma logico.

I linguaggi logici, la cui relazione di provabilita' preserva questa interpretazione dei connettivi logici come operazioni di ricerca (ossia soddisfa le condizioni sopra citate), vengono chiamati Abstract Logic Programming Languages (ALPL). Il linguaggio delle clausole di Horn con la provabilita' classica e' un ALPL; esso e' tuttavia un linguaggio molto "povero", nel senso che le operazioni di ricerca OR, AUGMENT, INSTANCE e GENERIC non sono usate. Analogamente, l'estensione higher-order alla logica delle clausole di Horn (con la provabilita' classica) e' un ALPL, anch'esso piuttosto debole in quanto non incorpora le operazioni di ricerca AUGMENT e GENERIC. Introdurre l'operazione AUGMENT richiede di passare dalla logica classica a quella intuizionista. Infatti la logica intuizionista fornisce un significato dichiarativo all'implicazione che e' consistente con quello dell'operazione di ricerca AUGMENT. Conseguentemente anche la classe delle hereditary Harrop formulas con la provabilita' intuizionistica e' un ALPL. E' inoltre possibile definire una versione higher-order delle hereditary Harrop formulas che combina le estensioni quantificazionali delle higher-order Horn clauses con quelle proposizionali delle first-order hereditary Harrop formulas. Il linguaggio che si ottiene, quello delle higher-order hereditary Harrop formulas con la provabilita' intuizionista, e' fra tutti l'ALPL piu' espressivo. Su di esso si basa il λProlog.

Dal punto di vista del linguaggio di programmazione, le estensioni quantificazionali e proposizionali della teoria logica supportano vari meccanismi di astrazione: i λ-termini e le variabli predicative permettono la programmazione higher-order; la possibilita' di avere implicazioni nei goal e nei body delle clausole (assieme al fatto che e' soddistatta la condizione AUGMENT per la derivabilita' operazionale) rende possibile la definizione di moduli; infine, la presenza dei quantificatori nei goal e nei body delle clausole fornisce un meccanismo per definire tipi di dato astratti.

Per quanto riguarda il supporto alla programmazione modulare, la implicazione nei goal puo' essere usata per strutturare, secondo una disciplina a stack, gli ambienti entro cui i goal sono dimostrati. Ad esempio, si consideri il goal
$(D1 \supset (G1 \wedge (D2 \supset G2))) \wedge G3$
nel contesto del programma P. Nel provare questo goal, tre differenti ambienti sono usati per dimostrare i sottogoal G1, G2 e G3: $P \cup \{D1\}$ per G1, $P \cup \{D1,D2\}$ per G2 e P per G3.

Per quanto concerne invece l'estensione higher-order, la presenza dei λ-termini e delle variabili predicative e funzionali fanno del λProlog un potente metalinguaggio, utile, ad esempio, per implementare "theorem provers", "program transformers" o per rappresentare la semantica del linguaggio naturale.

RELAZIONE SU 5TH JOINT CONFERENCE ON LOGIC PROGRAMMING - SEATTLE 88

P.G. BOSCO - CSELT - TORINO

Introduzione

Nel periodo 15-19 Agosto 1988 si e' tenuta a Seattle, WA, presso l'Universita' di Washington la "5th Joint Conference on Logic Programming" che quest'anno ha costituito l'unico evento internazionale sulla Programmazione Logica, "unificando" (bel termine!!) la normale Conferenza e il Symposium IEEE. Per questa ragione il numero di contributi sottomessi e dei partecipanti e' stato notevolmente alto: 250 contributi sono stati esaminati, di cui 102 selezionati; l'uditorio constava circa di 420 persone. Sfortunatamente, a causa del grande numero di presentazioni e tutorials tre o quattro sessioni erano mediamente attive in parallelo. Cio' mi ha impedito di avere una visione completa dei diversi filoni di ricerca e delle applicazioni di Programmazione Logica attualmente in corso nel mondo. Questo rapporto e' dunque limitato ai settori piu' specificamente di mio interesse. Nel seguito vengono presentati dei brevi commenti su tali aree di interesse. I riferimenti ad autori o gruppi di ricerca sono relativi agli articoli presenti nei Proceedings.

1. Esecuzione Parallela di Programmi Logici

Circa il 20% degli articoli e' inerente a modelli di esecuzione parallela di linguaggi logici. Alcuni di essi, [Raman], [Kale'] propongono nuovi modelli o nuove rappresentazioni delle sostituzioni in modo da superare le restrizioni usualmente imposte per la efficiente implementazione di modelli combinati AND-OR. Questi lavori, sebbene mostrino come le idee principali siano in corso di implementazione, non forniscono analisi quantitative, basate su osservazioni sperimentali, delle prestazioni globali ottenibili.

L'articolo di Carlsson presenta, anche se in forma piuttosto prototipale, come nel contesto di semplice OR-parallelismo (vedi Prolog parallelo di Argonne o del progetto Gigalips) "utili" forme di parallelismo AND possono essere ottenute attraverso la traduzione sintattica in costrutti OR-parallel-setof, alternativamente all'introduzione di piu' complesse macchine virtuali basate su meccanismi di join implementati piu' a basso livello (come nel modello PepSys dell' ECRC) Va detto che tale approccio e' stato precedentemente proposto, nell'87, da CSELT nel contesto

del progetto ESPRIT 415, come una ragionevole approssimazione ad un modello AND-OR per l'esecuzione parallela dei linguaggi logico-funzionali IDEAL e K-LEAF.

Anche nel semplice caso del parallelismo OR, alcuni modelli, alternativi al modello SRI di Warren sono stati studiati nell'ottica di strutture completamente distribuite. Essi sono basati su combinazioni di copia e ricomputazione. E' questo il caso della BC-Machine sviluppata al SICS e del modello DELPHI sviluppato all'SRI particolarmente orientato a problemi con forte preponderanza di computazione di ricerca. Le prime simulazioni mostrano come tali schemi basati sulla ricomputazione parziale, piuttosto che sullo sharing degli ambienti, non siano cosi' inefficienti come potrebbe sembrare a prima vista.

Altri progetti di ricerca sembrano aver raggiunto uno stadio implementativo sufficiente a produrre le prime analisi quantitative sperimentali del grado di parallelismo sfruttabile da configurazioni multiprocessori. Tra questi la ANLWAM (Argonne) e il PepSys (ECRC) dimostrano uno speed-up quasi lineare per problemi tipici di ricerca e parallelismo controllato di media granularita'.

La ricerca sui linguaggi "Committed Choice" e' ancora vivace. Molti articoli si soffermano sugli aspetti implementativi di tali linguaggi. Alkalai e Shapiro presentano la struttura di una architettura fisica per il Flat Concurrent Prolog basata su una rete di unita' computative, composte ognuna da quattro processori. La simulazione di tali nodi e' stata effettuata in FCP stesso. Una esperienza interessante e' stata portata avanti da Nilsson e Tanaka, sfruttando tecniche di vettorizzazione per l'implementazione del GHC sul supercomputer Hitachi S-820/80, ottenendo circa 0.5M riduzioni/sec. Klinger e Shapiro hanno inoltre sviluppato nuove tecniche di compilazione per l'FCP per ridurre il numero di riduzioni necessarie. Il risultante codice per certi programmi e' risultato piu' veloce del Quintus.

Come commento generale sul parallelismo si puo' dire che il parallelismo OR e' ad un buono stadio investigazione e sembra maturo per essere praticamente usato in applicazioni reali, mentre le combinazioni AND-OR soffrono maggiormente della complessita' imposta alle macchine virtuali sottostanti. Comunque proposte ragionevoli (come quella di Carlsson o la nostra) sono gia' presenti e sembrano efficaci per talune applicazioni. L'alternativa giapponese (cioe' di compilare Prolog in GHC e costruire macchine GHC) non ci sembra cosi' fortemente supportata da risultati resi disponibili alla comunita' scientifica.

2. Programmazione Logica e Funzionale

Parallelamente a lavori di raffinamento teorico dei meccanismi computazionali astratti, esiste un crescente interesse nell'uso del paradigma funzionale come un modo di interfacciare a certi livelli di astrazione, parti di programma che potrebbero essere implementati con linguaggi imperativi convenzionali. Due lavori [Bonnier, Zachary] affrontano questo problema, rilassando il requisito di completezza usualmente imposto all'unificazione equazionale per ottenere una maggiore efficienza su classi ristrette di programmi.

Il lavoro piu' interessante nel contesto logico+funzionale e' rappresentato dal Lambda-Prolog [Miller], un linguaggio di ordine superiore simile per certi aspetti a IDEAL, sviluppato in CSELT e presentato al Gulp87. Nel Lambda-Prolog si assume che l'unificazione di ordine superiore (nel lambda calcolo tipato) sia usata come meccanismo di base. Cio' da' la possibilita' di esprimere concisamente trasformazioni di programmi come procedure del linguaggio. Una implementazione efficiente non e' ancora disponibile e forse qualche restrizione della potenza del linguaggio sara' necessaria per un suo utilizzo in un contesto di "normale" programmazione.

3. Applicazioni

Anche questa volta come gia' nelle passate edizioni, notiamo che lo spazio dedicato alle applicazioni e' veramente esiguo (6% ; 10% se consideriamo anche il campo del linguaggio naturale). Cio' potrebbe derivare dal fatto che il comitato organizzatore e' largamente composto da persone provenienti da ambienti accademici e di ricerca. Inoltre, anche tra i lavori ammessi, soltanto pochissimi danno la sensazione dell'applicabilita' della programmazione logica in ambienti non tremendamente specialistici. .

Il piu' significativo, sotto questo aspetto, (presentato da Reintjes), riguarda un complesso sistema per la progettazione VLSI, il quale e' stato completamente riimplementato in Prolog con un risparmio in termini di linee di programma del 90% (da 100000 linee C a 10.000 Prolog) ed un miglioramento in termini funzionalita' offerte. L'efficienza del sistema e' stata giudicata soddisfacente per la maggior parte dei moduli, principalmente gli editors grafici.

Una interessante applicazione di CLP, un'estensione della programmazione logica con risolutori speciali di vincoli, e' stata presentata da Dincbas dell'ECRC che ha dimostrato come un noto problema di ricerca operativa puo' essere semplicemente programmato in CLP in modo piu' efficiente che con linguaggi tradizionali.

Il nostro contributo era nella stessa linea dei precedenti: cercava cioe' di propagandare l'idea che i nuovi linguaggi dichiarativi emergenti (nel nostro caso IDEAL, un linguaggio di ordine superiore logico + funzionale) non sono soltanto utili per applicazioni classificate di "intelligenza artificiale", ma possono, in ambienti di programmazione piu' convenzionale, essere usati proficuamente come linguaggi di specifica eseguibile, ed, in taluni casi, senza "tremendi" cali di prestazione rispetto a linguaggi convenzionali.

Nel campo del linguaggio naturale e' da segnalare un lavoro molto serio portato avanti da un gruppo dell' UNISYS [Hirschman,...] che ha verificato l'impatto dell'OR-parallelismo in un algoritmo di parsing realistico. E' stato adottato un modello all-solution parallelo, implementato da processi non comunicanti con condivisione di ambienti su un multiprocessore a memoria comune. La grammatica di circa 300 regole BNF copriva una buona parte dell'inglese. Per frasi relativamente complesse la velocizzazione dell'algoritmo e' stata dimostrata essere superiore alla meta' dei processori utilizzati.
Altri articoli confermano la bonta' dei linguaggi logici come linguaggi di implementazione per un ampio spettro di grammatiche e relativi parsificatori.

4. Tipi e Oggetti

L'introduzione di tipi polimorfici nella programmazione logica e' un modo per meglio costringere i programmi a conformarsi ai domini di appplicazione che l'utente ha in mente, attraverso una metodologia di prevenzione dell'errore a tempo di compilazione, che allo stesso tempo lascia la possibilita' di un riutilizzo dello stesso codice in differenti contesti.
Reddy discute possibili sistemi di tipi per il Prolog basati o solo su tipi universali, o su combinazioni di tipi esistenziali e universali.
Xu presenta un sistema di tipi, gia' prototipato, capace di modellare aspetti di polimorfismo parametrico, polimorfismo gerarchico e multiplo la cui semantica puo' essere data in uno stile tipico (Apt-VanEmden) della programmazione logica.
Voda da' una caratterizzazione semantica informale della struttura di tipi di Trilogy, un linguaggio che mischia Prolog, Lisp e Pascal ed offre algoritmi built-in per il soddisfacimento di vincoli.

La comunita' di programmazione logica sembra tuttora interessata ad incorporare concetti objet-oriented in Prolog. Tra i lavori relativi a questo argomento ricordiamo [Chen] che presenta una

estensione basata su particolari variabili che spaziano su domini di oggetti intesi come storie e [Conery] che, piu' praticamente, adotta la visione degli oggetti come collezioni di clausole e fa uso di clausole di Horn generalizzate (con teste multiple) per modellare le operazioni sugli oggetti. .

5. Interpretazione Astratta

L'interpretazione astratta accoppiata con la valutazione parziale e' riconosciuta come una efficace tecnica compilativa, che produce, dato un programma logico, una sua versione arricchita con informazioni addizionali estratte da tale procedimento, che possono essere sfruttate per ottenere in generale una maggiore efficienza. In particolare, gli articoli alla conferenza si sono concentrati sulla derivazione automatica di stati di istanziazione [Janssens] allo scopo di semplificare il meccanismo di unificazione, e di dipendenze tra variabili, per massimizzare il grado di parallelismo AND indipendente [Warren, Debray,Winsborough]. Il costo a tempo di compilazione introdotto dall'interpretazione astratta sembra giustificato dall'incremento di efficienza ottenibile a tempo di esecuzione.

6. Macchine Prolog Sequenziali

I ricercatori dell'Hitachi hanno presentato gli aspetti piu' significativi del loro processore Prolog capace di ottenere circa 1300 Klips. La macchinae' stata derivata da un processore preesistente, con l'aggiunta di circa un 3% di hardware specializzato, mirato soprattutto a rendere piu' efficiente la gestione del pipeline.

McCabe, dell'Imperial College, ha progettato un processore RISC chiamato SWIFT per l'esecuzione efficiente di LISP a PROLOG. Le simulazioni mostrano come prestazioni dell'ordine di 800Klips siano ottenibili con un clock a 20MHz.

Motorola e Applied Logic Programming hanno dimostrato una implementazione della WAM sul processore Motorola 88000. Le prestazioni osservabili erano di circa 450Klips (naive reverse) con liste interamente contenute nella cache e di 150 Klips con liste di grossa dimensione.

Ricercatori della Syracuse University e dell'Honeywell SRC hanno affrontato il problema della parallelizzazione dell'unificazione utilizzando memorie CAM. I risultati stimati sono dell'ordine di 0.6 - 5M unificazioni per secondo. Cio' tradotto in Lips dovrebbe corrispondere, secondo i

loro calcoli, a circa 1MLips ottenibile in modo interpretativo, cioe' senza ricorrere ad una compilazione del tipo Prolog -> WAM.

Commenti Generali

La mia sensazione e' che il comitato di programma avrebbe dovuto essere piu' selettivo nel giudicare i contributi. Cio' avrebbe sollevato il livello della conferenza e, allo stesso tempo, permesso ai partecipanti di avere una piu' ampia veduta sui reali avanzamenti ottenuti nel campo della programmazione logica, senza essere disorientati da interventi di dubbio valore scientifico.

Rispetto agli articoli presentati ed anche sulla scorta degli scambi di opinione avuti durante la conferenza, si puo' dire che il contributo europeo (ed italiano) e' sicuramente al livello dello stato dell'arte (ed anche piu'avanzato) per quanto riguarda gli aspetti teorici ed anche per taluni aspetti tecnologici. In particolare va segnalato lo sforzo globale dell'ECRC, dalle cui presentazioni traspare un serio impegno nel mantenere collegate una tecnologia Prolog d'avanguardia (anche hardware) con le esigenze dettate dalle applicazioni reali.

Soci Ordinari 1988

Francesco Abruzzese
NOOS
Via Fuorigrotta 26
80125 Napoli NA

Giambattista Amati
Fondazione U. Bordoni
Via B. Castiglione 59
00142 Roma RM

Flavio Argentesi
CEC CCR ISPRA
Via Fermi
21020 ISPRA VA

Elena Baraldi
ENIDATA spa
Via Medici del Vascello 36
20138 Milano MI

Roberto Barbuti
Dip. Informatica
Corso Italia 40
56125 Pisa PI

Giampiero Battocchioni
CONTRAVES Italiana
Via Affile 102
00131 Roma RM

Maurizio Benedetti
Epsilon
Paravia spa
Corso Racconigi 16
10139 Torino TO

Anna Maria Benzoni
Dip. Informatica e Sistemistica
Via Buonarrori 12
00185 Roma RM

Paola Bertaina
Dip. Informatica
Corso Svizzera 185
10149 Torino TO

Ilio Bocci
SIELTE
Via Campo Romano 71
00173 Roma RM

Fabio Agrò
ITALSOFT
Corso Rinascimento 52
00186 Roma RM

Francesca Arcelli
Dip. Scienze Informazione
Via Moretto da Brescia 9
20133 Milano MI

Patrizia Asirelli
IEI - CNR
Via S. Maria 46
56126 Pisa PI

Stefano Baratella
Dip. Matem. Pura e Applic.
Via Belzoni 7
35131 Padova PD

Carla Basili
ISRDS CNR
Via de Lollis 12
00185 Roma RM

Fabio Benedetti
Via Mantova 34
00100 Roma RM

Alfonso Benevento
Via Cavone
89047 Roccella Ionica RC

Francesco Bergadano
Dip. Informatica
Corso Svizzera 185
10149 Torini TO

Mario Bianchi
Via Rossini 93
0041 Albano RM

Paola Bomboi
Via B. Molinari 15
00194 Roma Rm

Luciana Bordoni
ENEA
S.P. Anguillarese km. 1,300
00060 Roma RM

Enrico Botto
Olivetti GSRI - DMR
Via C. Colombo 49
20090 Trezzano s/N MI

Daniele Briatico
INTECS Sistemi
Via Vertunno 2c
00157 Roma RM

Massimo Bruschi
Viale Matteotti 275
20099 Sesto S.Giovanni MI

Massimo Buonanno
Via Firenze 78
03100 Frosinone FR

Giuliano Calabrese
HITEC
Via Carmine 4
82010 S.Nazaro BN

Mauro Canarecci
ENIDATA spa
Via Medici del Vascello 26
20138 Milano MI

Alessandra Carbone
Dip. Matematica
Via del Capitano 15
53100 Siena SI

Maria Grazia Carrer
Via A. Inganni 84
20147 Milano MI

Gianluca Casale
Via G. Brodolini 32
00043 Roma RM

Francesco Cenci
Via A. Serpieri
00197 Roma RM

Annalisa Bossi
Ist. Algebra e Geometria
Via Belzoni 7
35100 Padova PD

Paolo Bresciani
I.R.S.T.
38050 Povo TN

Antonio Brogi
Dip. Informatica
Corso Italia 40
56125 Pisa PI

Michele Bugliesi
ENIDATA spa
Via Medici del Vascello 26
20138 Milano MI

Riccardo Burlon
TECSIEL
Via S. Maria 19
56126 Pisa PI

Filippo Calabresi
TECSIEL spa
Via Barnaba Oriani 32
00197 Roma RM

Amedeo Cappelli
ILC - CNR
Via della Faggiola 32
56126 Pisa PI

Bruno Cardile
TECSIEL spa
Via S. Maria 19
56126 Pisa PI

Giorgio Casadei
Dip. Statistica
Via Belle Arti 41
40126 Bologna BO

Marco Cavalieri
DEIS
Viale Risorgimento 2
40136 Bologna BO

Pietro Cenciarelli
TECSIEL
Via Oriani 32
00100 Roma

Franco Cerafogli
Via Calcagnodoro 11
02100 Rieti RI

Paola Chelli
Via Ammiraglio Marzolo 34
00122 Roma RM

Nicoletta Cocco
Dip. Matematica Pura e Appl.
Via Belzoni 7
35100 Padova PD

Attilio Colagrossi
IASI CNR
Viale Manzoni 30
00185 Roma RM

Raoul Collepiccolo
SOI Informatica srl
Via Fratelli Savio 3b
10121 Torino TO

Mirella Conti
TECSIEL
Via S. Maria 19
56126 Pisa PI

Patrizia Coscia
S & M
Via Edison 3/b
56017 S. Giuliano Terme PI

Stefania Costantini
Dip. Scienze Informazione
Via Moretto da Brescia
20133 Milano MI

Antonino D'Amico
Banco di S. Spirito
Via Padre Semeria 9
00154 Roma RM

Michela Degl'Innocenti
S & M spa
Vicolo S. Pierino 4
56127 Pisa PI

Stefano Cerri
Dida.el
Via Asmara 25
00199 Roma RM

Loretta Chicchiello
Dip. Fisica
Piazzale Tecchio
80100 Napoli NA

Annibale Cois
Via Puccini 32
09012 Capoterra CA

Livio Colussi
Dip. Matematica Pura e Appl.
Via Belzoni 7
35100 Padova PD

Luca Console
Dip. Informatica
Corso Svizzera 185
10149 Torino TO

Luigi Cordaro
Via delle Canarie 13
00121 Roma RM

Massimo Costa
Via Mori 29
16039 Sestri Lev. GE

Giovanni Criscuolo
Dip. Scienze Fisiche
Mostra D'Oltremare pad.19
80125 Napoli NA

Massimialiano De Angelis
Via Fermi 59
00044 Frascati RM

Carlo Del Gracco
ENIDATA
Via Chianesi 110
00128 Roma RM

Barbara Demo
Dip. Informatica
Corso Svizzera 185
10149 Torino TO

Giuseppe Di Battista
Dip. Informatica
Via Buonarroti 12
00185 Roma RM

Furio Di Paola
Dip. Sociologia
Via Torino 95
00185 Roma RM

Francesco Donini
Dip. Informatica e Sistemistica
00185 Roma RM

Alessandro Drago
Via Monti di Primavalle 194
00167 Roma RM

Susi Dulli
Dip. matematica Pura e Appl:
Via Belzoni 7
35100 Padova PD

Moreno Falaschi
Dip. Informatica
Corso Italia 40
56125 Pisa PI

Patrizia Ferranti
Texas Instruments
V.le delle Scienze
02100 Rieti RI

Maria Grazia Filippini
Via Foscolo 1
25016 Ghedi BS

Paola Forcheri
IMA CNR
Via L.B. Alberti 4
16132 Genova GE

Flora Frattini
Corso Libertà 52
20031 Cesano Maderno MI

Angelo Fucci
Via O. da Gubbio 199
00146 Roma RM

Giuliana Dettori
IMA - CNR
Via A. Ravà 124
00142 Roma RM

Andrea Di Carlo
Fondazione U. Bordoni
V.le Trastevere 108
00153 Roma RM

Giovanna Dondossola
Corso Bergamo 13
22053 Lecco CO

Carlo Donzella
Via Amendola 13
40100 Bologna BO

Vilma Draperi
Via Cesare Battisti 15
10129 Torino TO

Carlo Ellena
Database Informatica
Viale Umanesimo 32
00144 Roma RM

Laura Fanti
TECSIEL
Via S. Maria 19
56126 Pisa PI

Gilberto Filè
Istituto di Matematica
Via Belzoni 7
35100 Padova PD

Gisele Fischer Servi
Via Bachelet 2
43100 Parma PR

Sergio Fortini
SIP DG Roma
Via Torre Rossa 66
00165 Roma RM

Giovanni Frigerio
Via Masaccio 41
50132 Firenze FI

Celeste Gallo
CONSOFT
Via Gioberti 78
10128 Torino TO

Gianpiero Garavagno
Sipe Optimation spa
Via Campo Ascolano 33
00040 Pratica di Mare RM

Marcello Giacomantonio
TE.COM srl
P.zza Matteotti 1
61040 Mondavio PS

Laura Gianetto
ESACONTROL
Via Hermada 6
16100 Genova GE

Laura Giordano
Dip. Informatica
Corso Svizzera 185
10149 Torino TO

Roberto Gorrieri
Dip. Informatica
Corso Italia 40
56125 Pisa PI

Sergio Greco
CRAI
Località S. Stefano
87036 Rende CS

Paolo Guidotti
IDG
Via Panciatichi 56/16
50127 Firenze FI

Gaetano Lanzarone
Dip. Scienze dell'Informazione
Via Moretto da Brescia 9
20133 Milano MI

Maurizio Lenzerini
Dip. Informatica e Sistemistica
Via Buonarroti 12
00185 Roma RM

Giorgio Levi
Dip. Informatica
Corso Italia 40
56125 Pisa PI

Carla Limongielli
Via Acqui 18
00183 Roma RM

Mauro Gaspari
Delphi spa
Via della Vetraia 11
55049 Viareggio LU

Sabrina Gianpieri
Via Grumo Appula 15
00133 Roma RM

Giancarlo Giardina
Via Teramo 37
65121 Pescara PE

Elio Giovannetti
Via Cordero di Pamparato 28
10148 Torino TO

Roberto Grande
Dip. Informatica
Via Moretto da Brescia 9
20133 Milano MI

Gaetano Guardasole
Univers. Calabria
Dip. Sistemi
87026 Arcavata di Rende CS

Evelina Lamma
DEIS
Viale Risorgimento 2
40136 Bologna BO

Riccardo Laurenti
Via Filipponi 14
00100 Roma RM

Pierluigi Lettieri
CONTRAVES
Via Affile 102
00131 Roma RM

Francesco Licciardi
HITEC
Viale delle Ginestre 52
84100 Salerno SA

Roberto Lofaro
C.P. 174
10100 Torino TO

Luciano Lucchesi
Sipe Optimation
Vicolo S. Pierino 4
56127 Pisa PI

Federico Macchi
DIGITAL spa
Via Fortezza 11
20126 Milano MI

Bruno Magnani
Sipe Optimation
Lab. Ricerca e Sviluppo
Via Campo Ascolano 33
00040 Pratica di Mare RM

Stefano Maiolatesi
Viale I Maggio 9
00046 Grottaferrata RM

Cinzia Marchegiani
Via di Vermicino 563
00044 Roma RM

Francesco Mariani
CARISMA spa
Via F. Baracca 5a
06100 Perugia PG

Alberto Martelli
Dip. Informatica
Corso Svizzera 185
10149 Torino TO

Anna Martinoli
Elsag
Via Puccini 2
16100 Genova GE

Marco Mascardi
Elsag spa
Via Puccini 2
16100 Genova GE

Maurizio Mazzer
CSM
Via di Castel Romano 100
00129 Castel Romano RM

Monica Lugaresi
ENIDATA spa
Via Medici del Vascello 26
20138 Milano MI

Anna Maglia
Via delle Azzalee 21
Appiano G. CO

Tina Magro
Tecnologica
Via del Serafico 75
00142 Roma RM

Paolo Mancarella
Dip. Informatica
Corso Italia 40
56125 Pisa PI

Luigi Marcolungo
ISI
Via del Santo 28
35100 Padova PD

Giorgio Marotta
Database Informatica
Viale Umanesimo 32
00144 Torino TO

Maurizio Martelli
CNUCE CNR
Via S. Maria 56
56126 Pisa PI

Umberto Marzaroli
Univ. di Bologna
Via Belle Arti 41
40126 Bologna BO

Giovanni Mascari
IAC CNR
V.le del Policlinico 137
00161 Roma RM

Francesco Mazzuca
TECOM
P.zza Matteotti 1
61040 Mondavio PS

Roberto Melchiori
CEDE
Villa Falconieri
00044 Frascati RM

Gian Carlo Meloni
Univ. Milano
Via C. Saldini
20133 Milano

Vitaliano Milanese
Dip. Matematica Informatica
Via Zanon 6
33100 Udine UD

Giancarlo Minicone
Via Marconi 76
80046 S. Giorgio a CR. NA

Alessandro Mirannalti
Bibl. Document. Pedagogica
Via Buonarroti 10
50122 Firenze FI

Corrado Modoni
IBM Italia
Passaggio Gaudenzio 1
35100 Padova PD

Alberto Molesini
Via Doninzetti
37060 Sona VR

Ugo Montanari
Dip. Informatica
Corso Italia 40
56125 Pisa PI

Lorenzo Moretti
ILC CNR
Via della Faggiola 132
56126 Pisa PI

Ugo Moscato
Istituto Cibernetica
Via Moretto da Brescia 9
20133 Milano MI

Alessandro Mura
Selenia
Via Bergamini 50
00159 Roma RM

Paola Mello
DEIS
V.le Risorgimento 2
40136 Bologna BO

Luisa Mich
Istituto Informatica
Via Verdi 26
38100 Trento TN

Gaetano Minciullo
Tecnologica
Via del Serafico 75
00142 Roma RM

Eliana Minicozzi
Dip. Informatica Sist.
Via Claudio 21

Claudio Mirolo
Dip. Matematica e Informatica
Via Zanon 6
33100 Udine UD

Maria Teresa Molfino
IMA CNR
Via L. B. Alberti 4
16132 Genova GE

Milena Monduzzi
Via G. M. Angioy 34
09100 Cagliari CA

Stefano Montesi
SYSMEDA
Piazzale Clodio 14
00195 Roma RM

Giuseppe Mosca
Via Sgurgola 13
00179 Roma RM

Elio Mungo
Via Zacchia 11
00161 Roma RM

Liliana Murino
Database Informatica
Viale Umanesimo 32
00144 Roma RM

Umberto Nanni
Dip. Informatica
Via Buonarroti 12
00185 Roma Rm

Teresa Napoli
Viale Firenze 80
85100 Potenza PZ

Daniele Nardi
Dip. Informatica
Via Buonarroti 12
00185 Roma RM

Gianluca Nebiacolombo
Esacontrol spa
Via Hermada 6
16156 Genova GE

Simone Nodari
Via Perugia 52
00043 Ciampino RM

Maria Vittoria Oneto
ELSAG
Via Puccini 2
16154 Genova GE

Enrico Pagello
LADSEB CNR
Corso Stati Uniti 4
35100 Padova PD

Catiuscia Palamidessi
Dip. Informatica
Corso Italia 40
56125 Pisa Pi

Livio Palomba
Via Camelie 8
20147 Milano MI

Giovanni Panti
Via Cappuccini 128
53100 Siena SI

Federico Patricelli
S.S. Reiss Romoli
S.S. per Coppito km. 0,300
67100 L'Aquila AQ

Roberto Nannucci
IDG CNR
Via Panciatichi 56/16
50177 Firenze FI

Domenica Nardelli
Via Catania 3
85100 Potenza PZ

Antonio Natali
DEIS
V.le Risorgimento 2
40136 Bologna BO

Andrea Nicoli
Via D'Azeglio 4
56125 Pisa PI

Carmela Nolè
Via Sabbioneta 9
85100 Potenza PZ

Mario Ornaghi
Dip. Scienze Informazione
Via Moretto da Brescia 9
20133 Milano MI

Paolo Pagliari
Frosinone-Casalvieri
03034 Casalvieri FR

Aldopaolo Palareti
Ist. Informatica
Via Brecce Bianche
60100 Ancona AN

Massimo Paltrimieri
Honeywell Bull Italia
Via Vida 11
20127 Milano MI

Francesco Pastore
Automata srl
Via Bertolotti 7
10121 Torino TO

Maria Teresa Pazienza
Dip. Informatica Sistemistica
Via Buonarroti
00185 Roma RM

Antonella Pelaggi
SIP - DG
Via di Val Cannuta
00166 Roma RM

Donatella Persico
ITD CNR
Via Opera Pia 11
16145 Genova GE

Alberto Pettorossi
IASI CNR
Viale Manzoni 30
00185 Roma RM

Renato Picco
CEDE
Villa Falconieri
00044 Frascati RM

Salvatore Pluchino
Dip. Matematica
Viale A. Doria 6
95125 Catania CT

Maurizio Pozzoni
Banca Popolare di Sondrio
Piazza Garibaldi 16
23100 Sondrio SO

Giuseppe Quintarelli
Via S. Vito 36b
37024 Negrar VE

Claudia Roda
CARISMA
Via Baracca 5a
06100 Perugia PG

Gianfranco Rossi
Dip. di Informatica
Corso Svizzera 185
10149 Torino TO

Adolfo Russo
Via G. Silvati 32
80141 Napoli NA

Giovanni Santi
Via Cappuccini 128
53100 Siena Si

Barbara Pelletti
Tecnologica
Via del Serafico 75
00142 Roma RM

Silvio Pes
Via Turati 23
07100 Sassari SS

Flavia Piazza
Informatica 2
Via Vassalli 2
10138 Torino TO

Clara Pizzuti
CRAI
Località S. Stefano
87036 Rende CS

Massimo Poesio
Computer Science Dept.
P.O. Box 302762
2000 HH 6 Hamburg
Germania

Maurizio Proietti
IASI CNR
V.le Manzoni 30
00185 Roma RM

Andrea Ripanucci
Tecsiel
Via B. Oriano 32
00100 Roma RM

Francesca Rossi
Dip. Informatica
Corso Italia 40
56125 Pisa PI

Cristina Ruggieri
Enidata
Via Medici del Vascello 26
20138 Milano MI

Francesco Sacerdoti
NOOS snc
Via Parco Margherita 23
80121 Napoli NA

Giuseppe Sardu
S & M spa
Vicolo S. Pierino 4
56127 Pisa PI

Luigi Sarti
ITD CNR
Via Opera Pia 11
16145 Genova GE

Marco Schaerf
Via della Rotonda 4
00186 Roma RM

Mario Servi
Dip. Matematica
Via Bachelet 2
43100 Parma PR

Romeo Sperandio
Telespazio spa
Via Bergamini 50
00159 Roma RM

Fabio Tarini
CNUCE CNR
Via S. Maria 56
56126 Pisa PI

Lea Terracini
Corso Dante 118
10126 Torino TO

Gaetano Trainito
Ladseb CNR
Corso Stati Uniti 4
35020 Padova PD

Aldo Ursini
Dip. Matematica
Via del Capitano 15
53100 Siena SI

Giuliano Valeriani
Carisma
Via Baracca 5a
06100 Perugia PG

Antonella Vecchio
Palazzo Alitalia
P.le G. Pastore 6
00144 Roma EUR RM

Dario Sartini
SISMEDA
P.le Clodio 14
00195 Roma RM

Biagio Scognamiglio
CRAI
Via Tevere 15
00198 Roma RM

Maria Simi
Dip. Informatica
Corso italia 40
56125 Pisa PI

Stefano Stefani
Dip. Chimica Fisica
Univ. di Venezia
Calle Larga S. Marta 2137
30123 Venezia VE

Paolo Terenziani
Dip. Informatica
Corso Svizzera 185
10149 Torino TO

Luigia Torre
S & M
Via Andrea Pisano 7
56122 Pisa PI 10126

Daniele Turi
Dip. Informatica
Corso Italia 40
56125 Pisa PI

Gigliola Vaglini
Dip. Informatica
Corso Italia 40
56125 Pisa PI

Maurizio Valle
Via Piave 29
17025 Loano SV

Antonella Verdino
Via Bruzzesi 35
20146 Milano MI

Antonio Vincenzi
Via Belvedere 17/1
17012 Albisola Mare SV

Maurizio Zamboni
Dip. Elettronica
Corso Duca degli Abruzzi 24
10129 Torino TO

Sandra Zanobetti
Tecsiel
Via S. Maria 19
56126 Pisa Pi

Michele Volpe
Fondaz. U. Bordoni
V.le Europa 190
00144 Roma RM

Luigi Zanella
Via Bentivogli 19
40055 Castenaso BO

Andrea Zinno
Tecsiel
Via B. Oriani 32
00197 Roma RM

Soci Collettivi 1988

DIGITAL
Giorgio Berini
Via Testi 11
20092 Cinisello MI

CSI PIEMONTE
Daro de Jaco
Corso Unione Sovietica 216
10134 Torino TO

DATABASE INFORMATICA
Margherita Errico
V.le Umanesimo 32
00144 Roma RM

IBM ITALIA
Centro Scientifico IBM
Marcello Marelli
Via del Giorgione 159
00147 Roma RM

SIEMENS DATA
Carlo Monterino
Viale Monza 347
20126 Milano MI

SARIN SPA
Stefano Pavan
S.S. 148 Pontina Km. 29,1
00040 Pomezia RM

CRAI
Gaetano Santucci
Via Tevere 15
00198 Roma RM

KTI
Ivano Coltellacci
Via Monte Carmelo 5
00166 Roma RM

OTO MELARA
Ettore Durando
Via Valdilocchi 15
19100 La Spezia SP

OLIVETTI
Armando Limongiello
Via C. Colombo 49
20090 Trezzano/Naviglio MI

S & M
Mario Modesti
Vicolo S. Pierino 4
56127 Pisa PI

ENIDATA
Eugenio Omodeo
Via A. Moro 38
40100 Bologna BO

APPLE COMPUTERS
Roberto Poggio
Palazzo Q8 Milano Fiori
20089 Rozzano MI

ENIDATA SPA
Francesco Zambon
Div. Prodotti Tema
Via A. Moro 38
40127 Bologna BO

QUARTO CONVEGNO NAZIONALE SULLA PROGRAMMAZIONE LOGICA GULP '89

Il prossimo Convegno Nazionale sulla Programmazione Logica del GULP si terrà a Bologna presso la Facoltà di Ingegneria, dal 7 al 9 giugno , organizzato dal Dipartimento di Elettronica, Informatica e Sistemistica.

L'interesse dimostrato, le adesioni ed il numero di lavori sottoposti, fanno ben sperare anche per quest'anno, in una buona riuscita del convegno, ormai punto di incontro e discussione abituale per tutti coloro che in Italia si occupano di programmazione logica.

Il convegno sarà preceduto da due giornate di tutorials (5 e 6 giugno) strutturate in due differenti sessioni parallele per poter soddisfare esigenze eterogenee. Una prima sessione di tutorials sarà dedicata a coloro che si avvicinano per la prima volta alla programmazione logica e sarà organizzata in due parti complementari: la prima verterà sui fondamenti, la seconda su esempi e applicazioni.

La seconda sessione di tutorials, avanzati, approfondirà temi di carattere più specifico nell'ambito della programmazione logica quali la negazione, l'interpretazione astratta ed il parallelismo. È prevista inoltre, nell'ambito di queste prime giornate, una sessione dedicata alla programmazione logica e la didattica.

Le tre giornate del convegno prevedono, come di consueto, oltre alla presentazione dei contributi, tre relazioni invitate. I relatori invitati sono K.R. Apt (University of Texas at Austin) che terrà una relazione dal titolo "Efficient Computing of Least Fixpoint", D.H.D. Warren (University of Bristol) che terrà una relazione dal titolo "The Andorra Model for Combining and- and or- parallelism in Prolog" e L. Monteiro (University of Lisbon) che terrà una relazione sui linguaggi paralleli di programmazione logica con particolare riferimento al Delta-Prolog.

Stiamo inoltre organizzando una particolare sessione, durante il convegno, per presentare una panoramica dei progetti di ricerca sulla programmazione logica italiani ed internazionali.

Poiché abbiamo già ricevuto parecchie adesioni, pensiamo di poter organizzare un'esposizione di sistemi hardware e software particolarmente ricca ed interessante.

Paola Mello
DEIS, Università di Bologna.

Scheda di iscrizione da compilare e spedire a:
Giuliana Dettori, I.M.A - CNR, Via L.B. Alberti, 4, 16132 Genova

Il Sottoscritto/a

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

Ditta/Ente/Università.....

Posizione ricoperta.....

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

CAP.....Città.....Prov.....

Indirizzo al quale desidera ricevere la corrispondenza

- ❑ lavoro
- ❑ altro.....

.....

- ❑ chiede di essere iscritto al GULP - Gruppo Ricercatori e Utenti di Logic Programming,in qualità di socio ordinario.
- ❑ chiede l'iscrizione al GULP della Ditta/Ente.....
  in qualità di socio collettivo

Socio AI*IA ❑ si ❑ no

Ha versato la quota di iscrizione di lire .....mediante

- ❑ versamento su cc postale n. 11147568 intestato a GULP-Pisa
- ❑ assegno accluso intestato a GULP.

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

Quota annuale di iscrizione (comprendente l'iscrizione ALP):

Soci Ordinari	(solo soci GULP)	Lire	40.000
	(soci AI*IA)	Lire	35.000
Soci Collettivi		Lire	500.000
Soci Studenti	(senza iscrizione ALP)	Lire	20.000

Si prega di fornire tutte le informazioni richieste.