Information Technology: Research and Development (1983), *2* (233–238)

LOGIC PROGRAMMING WORKSHOP 83

P. F. WILK

Department of Artificial Intelligence, University of Edinburgh, UK

Over 80 delegates from 17 different countries attended the Logic Programming Workshop 83 in the Algarve, Portugal, from 26 June to 1 July. The workshop was organized by the Nucleo de Inteligencia Artificial, of Departmento de Informatica, Universidade Nova de Lisboa. The program chairman was Luis Moniz Pereira.

The workshop consisted of 43 formal presentations (35 of which appear in the proceedings) and six panel sessions. The sessions were divided into topics on: natural language; knowledge base systems; logic programming theory; Prolog implementation; databases and logic programming methodology.

Jan Chomicki (Warsaw University, Poland) discussed the problems related to using Prolog as an implementation language for databases. In particular the paper discusses how to organize and access large Prolog databases (based on extendible hashing and partial match retrieval).

Wlodzimierz Grudzinski (Warsaw University, Poland) described SPOQUEL–a query language for relational databases (written in Prolog).

Tomasz Pietrzykowski (Acadia University, Wolfville, Canada) presented a database model of a functional programming language, called PROGRAPH, which uses a graphical display for the user interface.

Luis Pereira (Universidade Nova de Lisboa, Portugal) presented a relational database modeller for generating databases. The program uses information gathered interactively, from the user, to generate specific menu-based consultation programs.

Jan Komorowski (Harvard University, USA) presented a universal display editor (not in the proceedings) as a software prototyping language tool. An example application of a Pascal syntax-directed editor was explained.

Patrick Saint-Dizier (IRISA, Universitaire de Beaulieu, France) described a way of building an intelligent interface between a human and a computer.

Antonio Porto (Universidade Nova de Lisboa, Portugal) gave a talk about the natural language interface to a garden store assistant (written in Prolog).

Miguel Filgueiras (Universidade Nova de Lisboa, Portugal) described the design of a kernel for a knowledge directed parser of natural languages. To check consistency of syntax analysis with respect to meaning, non-application dependent semantic tests are performed during syntax analysis. The application dependent parts of the semantic analysis are specified in a separate module. Therefore, it is claimed that it is easier to adapt the interface to new applications.

Paul Sabatier (University of Paris, France) presented a formalism and implementation technique by which left and/or right contextual constraints (used in contextual grammars to specify rule ordering) can be easily expressed and efficiently computed in Prolog II. The implementation technique builds a graph containing

0144-817X/83/03 0233-06 $03.00

contextual information (built during parsing) which may be used to recover the context when a contextual constraint has to be satisfied.

Veronica Dahl (Simon Fraser University, Canada) chaired the panel on natural language, the theme of which was current trends in logic grammars. A case was made for context-sensitivity in grammars contrary to the current trend of augmenting context-free grammars with new rules during the parse.

Martin Williams (Heriot-Watt University, UK) described the implementation of an approach to security and integrity in query-by-example based on the idea of maintaining the consistency of data in the database. The approach extends the conventional types of integrity constraint to include functional, multivalued and embedded multivalued dependencies.

Jose Neves (Heriot-Watt University, UK) presented an extension to query-by-example (written in Prolog) that enables a user of a database to obtain positive and negative feedback information from queries or updates that are incomplete or incorrect.

Igor Mozetic (Jozef Stefan Institute, Ljubljana, Yugoslavia) described the development of an expert system that models the electrical behaviour of the heart. The model is used to automatically generate a knowledge base of all physiologically possible combinations of cardiac arrhythmias and their corresponding ECG descriptions.

Ferenc Darvas (Szki, Budapest, Hungary) described a logic-based expert system for model building in regression analysis. The system (written in M Prolog and FORTRAN, which communicate by files) has been used to test drug design performance.

Eugenio de Oliveira (Universidade Nova de Lisboa, Portugal) described a proposal to develop expert system building tools in Prolog. The system will combine a knowledge base acquisition subsystem (gathering semantic nets, metaknowledge and production rules) with a consultation subsystem (which uses metaknowledge to guide the developer through the presentation of explanations, reasoning and deductions).

Ed Stabler (University of Western Ontario, Canada) discussed the problem of achieving optimally efficient response to queries addressed to a large deductive database. Moreover, where the user wishes to interactively optimize the query for subsequent use, the system permits the interactive addition of general rules (expressions containing logical variables) as well as particular facts (expressions containing no variables) to the database.

Jack Minker (University of Maryland, USA) chaired the panel on knowledge based systems. Current issues in developing expert systems were discussed, in particular: what distinguishes an expert system from an application program? how is the expert's knowledge acquired and represented? how are temporal data handled? what tool-kit should be provided for the expert system developer? what morals should be applied to expert systems? how is search controlled in an expert system and what are the criteria for user acceptability (particularly when different classes of user perceive different system models)?

Pierre Deransart (INRIA, France) proposed an operational algebraic semantics for Prolog programs that follows resolution.

Andrzej Lingas (Linkoping University, Sweden) proposed that in order to fully understand the behaviour of parallel goal execution of logic programs it was necessary to apply the ideas of Turing-machine complexity theory to the complexity measures of logic programs, i.e., goal size, goal length, goal depth and conjunctive goal size.

Dan Sanlin (The Royal Institute of Technology, Sweden) described an abstract

machine called 'gepr' (goal, environment, program and resumption register). The gepr machine is a state transition system. The paper contains a definition for the transition rules that convert one state to the next. Each rule corresponds to a rule in a natural deduction system.

Patrizia Asirelli (Istituto Elaborazione Informazione, Pisa, Italy) described a fixed-point semantics of horn clauses with infinite terms—infinite terms are often used to define parallel communicating processes in Prolog, but current semantic definitions are incomplete and apply to unwanted infinite terms. Two semantic definitions are proposed based on least fixed-point construction. A proposed operational semantic definition generates a unit clause—representing a terminal clause (if non has been defined). A fixed-point semantics is defined which reflects the idea that non-terminal symbols are partial approximations of infinite terms.

Patrizia Asirelli then commented on some aspects of the first order semantics of a connective suitable for expressing concurrency. This is achieved by introducing the concept of class–a cluster of concurrent atoms–where an atom has a predicate and 'N' terms. Processes communicate by semaphores manifested as shared logical variables.

Maarten van Emden (Imperial College, London, UK) chaired the panel on Logic Programming Theory. The panel commented on the growth in theory papers. In particular many ideas had been transferred from other fields such as complexity theory. The topics discussed included complexity, concurrency, infinite terms, operational and fixed-point semantics and negation by failure. John Lloyd (Melbourne University, Australia) announced the release of MU–Prolog, which is written in 'C', has a correct implementation of negation by failure, database support, and a syntax similar to DEC–10 Prolog.

Gerard Ballieu (Katholieke Universiteit Leuven, Belgium) described a virtual machine to implement Prolog. The machine is based on David Warren's abstract Prolog machine but uses a structure copying technique for working storage.

David Bowen (Edinburgh University, UK) described a Prolog implementation, similar to that of Ballieu, that aims to combine a high degree of portability with speed and an efficient utilization of memory. The virtual machine for the implementation is written in the programming language 'C'. The design is well suited to optimization for particular machines, because there is a central core which can be translated into microcode or assembly language.

Paul Wilk (Edinburgh University, UK) described the production and evaluation of a set of Prolog benchmarks (not in the proceedings). The benchmarks consist of a number of large AI programs and over 100 small benchmarks. A methodology was described for producing the small benchmarks (each one of which is designed to benchmark a particular facet of an implementation) which can be applied to other AI languages. Benchmarking results were given for four Prolog implementations on six different computer systems.

Frank McCabe (Imperial College, London, UK) briefly described Lambda Prolog which is an attempt to unite Lambda Calculus and logic programming. He then described Abstract Prolog Machine (APM) which will be used as a target architecture for Lambda Prolog. However, APM is seen mainly as a Prolog architecture for single user machines.

Hiroshi Nishikawa (Institute for New Generation Computer Technology, Japan) gave an overview of the design of the Personal Sequential Inference Machine architecture. PSI has a 40 bit word format (8 bit tag and 32 bit data). It has a 32 bit real address space (no virtual addressing) with a non-structure sharing implementation of

working storage. The language system consists of a subset of DEC-10 Prolog with extended abilities for hardware resource handling, interrupt handling and process control. Notedly, the machine architecture includes provision for garbage collection and process switching.

Marco Bellia (Universita di Pisa, Italy) proposed a compiler that maps Prolog to a demand driven architecture. This is done by automatically annotating clauses (similar to automatic generation of mode declarations) according to functional dependencies. An annotation distinguishes between an atomic formula that computes a value for a given variable and one which uses the value of the variable.

Stanislaw Matwin (University of Ottawa, Canada) described an intelligent backtracking algorithm, applicable to first order logic. Essentially, a depth first search of the proof tree is directed by information kept in a separate graph structure, which represents the unifications generated during the proof.

Jack Minker outlined PRISM—A Parallel Inference System for Problem Solving. PRISM is based on logic programming and is implemented on ZMOB, a parallel multi-microprocessor system. The system is designed to provide a general experimental tool for the construction of large artificial intelligence problem solvers.

Seif Haridi (The Royal Institute of Technology, Sweden) described a mechanism that would control the traversal of the search tree in an Or-Parallel token machine (where a token pool contains processes that are ready to execute but have not yet been allocated a processor). This is complemented by a mechanism that prunes the search tree, removing branches that are obsolete computations.

Subsequently Haridi explained a machine architecture (similar to that of ALICE by John Darlington of Imperial College, UK) for the Or-Parallel token machine.

Maurice Bruynoogne (Katholieke Universiteit Leuven, Belgium) chaired the panel on Prolog implementation. The talk focused on the effect of a Prolog implementation on programming style–noting that programming elegance was often sacrificed for time and space efficiency. The desire for a Prolog standard was mooted because of Prolog portability problems. But generally, delegates thought that the subject area was at too early a stage in its evolutionary development to merit this.

E. Elcock (University of Western Ontario, Canada) gave reasons why Prolog should not be thought of as a specification language. In particular, the procedural semantics of a Prolog program are incomplete; often it is not clear that a program will terminate so it is necessary to consider proof of termination of a program.

Pavel Brazdil (Faculdade de Economia, Porto, Portugal) discussed the problems associated with the development of Prolog programs (not in the proceedings). Suggestions were made for a Prolog tool-kit similar to that of Interlisp.

Richard O'Keefe described a polymorphic type system for Prolog (obtainable from DAI, Hope Park Square, Edinburgh University, UK) and how it integrated with other Prolog development tools written at Edinburgh University. One advantage of a good type system is that it provides a static tool for determining whether all cases in a Prolog predicate have been considered. Moreover this type system can be used as a basis for encapsulation, providing an abstract data type facility.

The panel on databases was chaired by Jack Minker. The panel discussed Herve Gallaire's paper (Laboratoires de Marcoussis, France) 'Logic databases vs. deductive databases'. Gallaire's important paper presents a taxonomy of databases formulated by decomposing logic programming and databases into their component parts and studying their interconnections. Detailed descriptions of two important contrasting implementation approaches are described: logic databases and deductive databases. Logic databases are built above or aside Prolog and have their own description and

manipulation languages. The deductive database approach uses logic to provide extensions to conventional database systems, if only to remove the problem of implementing query languages with procedural languages. Gallaire expects that the Japanese Fifth Generation Project will reveal a more precise taxonomy than his, based on axioms rather than relationships.

Madhur Kohli (University of Maryland, USA) presented a theory for the intelligent control and execution of function free logic programs based on integrity constraints. The integrity constraints may be user supplied or automatically generated at run-time by analysis of goal failure.

Chris Moss (USA) defined a predicate 'seqof' that enables the total set of solutions to a problem to be returned individually, and processed individually, rather than returned collectively and processed as required.

Harvey Abramson (University of British Columbia, Canada) gave a definition of HASL (contained in the proceedings; written in Prolog). HASL is a purely applicative language which uses SASL's combinator reduction machine in conjunction with unification based conditional binding expressions.

Ed Sabb (ICL, Stevenage, UK) put forth an alternative philosophy for adapting resolution for logic programming termed finite computation principle. The principle maintains the power of symbolic substitution but seeks to manage infinite processes by combining order independence of predicate execution with infinite process detection. Management is performed by knowing and detecting the conditions that lead to infinite processes and then applying axioms of logic to determine a predicate order that makes the computation finite.

Keith Clark (Imperial College, London, UK) chronicled the historical development of logic programming in relation to computer science (not in the proceedings). From this chronology the reasons for the features incorporated into PARLOG (a parallel logic programming language) were rationalized. The main features of the language are: modules; and-parallelism; or-parallelism; eager functions; lazy functions; set expressions and Prolog as a subset (unlike concurrent Prolog).

Ehud Shapiro presented a rationale for the design of Concurrent Prolog. Of particular note was his reluctance to move away from simplicity until experimentation had confirmed him of the features that should be added to the language (not in the proceedings, a copy of the interpreter, written in Prolog, can be obtained from Department of Applied Mathematics, The Weizmann Institute of Science, Rehovot 76100, Israel).

Akikazu Takeuchi (Institute for New Generation Computer Technology, Japan) described interprocess communication in Concurrent Prolog. This is realized by sharing variables amongst processes. Therefore when a shared variable is instantiated to a message all processes sharing the variable receive the message. However, in Prolog, destructive assignment of shared variables is not permitted so every time a message is sent a new shared variable must be generated for the next communication. This form of communication is known as streaming.

Luis Monteiro (Universidade Nova de Lisboa, Portugal) described work, similar to that described by Asirelli, for concurrent programs. However, in this work Prolog is extended with the concept of an event, which gives a temporal logic programming language.

Antonio Porto (Universidade Nova de Lisboa, Portugal) described a concurrent language design that combines action reduction with logic programming. The main features of the language are: synchronization; concurrency; action rules and abstract data types.

The panel on logic programming methodology was chaired by Ehud Shapiro. The session was oriented to research methodology rather than programming methodology. (Delegates noted that there were no good texts available that described the programming methodology and techniques applicable to logic programming. However, Ehud Shapiro announced that shortly he would have a book published suitable for advanced logic programmers. Similarly, Luis Pereira will shortly have a book published for the naive logic programmer.)

It should be noted that in some cases presentation of work was not given by the author of the paper that appears in the proceedings. Furthermore, some papers that appeared in the proceedings were not presented at the conference and so are not covered here.

The following list represents the open research areas covered by the conference: (logic programming is defined here to be synonymous with Prolog)

1. Database implementation.
2. Natural language processing.
3. Query languages.
4. Intelligent user interface.
5. Expert systems.
6. Expert system building.
7. Logic programming semantics.
8. Parallel logic programming machines.
9. Sequential logic programming machines.
10. Abstract logic programming machines.
11. Intelligent proof-tree search.
12. Logic programming tool-kits.
13. Sequential logic programming languages.
14. Parallel logic programming languages.

Copies of the conference proceedings can be ordered by sending a personal cheque of 1700 Escudos (or equivalent) to Luis Moniz Pereira at the Nucleo de Inteligencia Artificial, of Departmento de Informatica, Universidade Nova de Lisboa, Quinta da Torre, Lisbon, Portugal. Selected papers will soon be published in book form.