Computer Science and Information Systems

Rekenaarwetenskap en Inligtingstelsels
Regardless of the size of your computer system it is a strategic tool. We are committed to helping you use it most effectively. All it needs is a little imagination.

UNIDATA
Telephone (012) 45-1152

THERE'S A GIANT IN EVERY ONE OF US
The Department of Computer Science at Pretoria University is inviting applications for two annually awarded bursaries for MSc study. These bursaries are aimed at outstanding students who have a Computer Science Honours degree (or equivalent qualification). As far as possible, one such bursary will be awarded to a South African student, while the other will be awarded to a student from another African state. The hope is to thereby promote interaction between South African computer scientists and computer scientists elsewhere in Africa.

The bursaries are funded by the Department's contract research work and will amount to a minimum of R20000 per year. They are intended to cover costs for tuition, housing, transport, books, and an adequate allowance for other living expenses. Successful candidates will be required to work for 20 hours per week as part of the Computer Science Department's contract research team. The work assigned to each candidate will be aimed at providing him with practical experience in his field of interest.

Study fields for which bursaries will be considered include:
- Computer Networking
- Distributed Systems
- Software Engineering
- Artificial Intelligence
- Intelligent Computer Aided Instruction
- Machine Aided Translation

Students should aim to complete the dissertation that is required for an MSc within two years of study (i.e. one year of full time study). The dissertation should preferably be written in English but it may also be written in Afrikaans.

The continuation of the bursary and adjustments to the allowance will be decided upon annually and will depend on progress. It is a condition of the bursary that, after completing his/her studies, the student should return to his/her country of origin for the same number of years for which the bursary was held.

Applicants may request application forms from:

   MSc. Bursaries
   The Computer Science Department
   University of Pretoria
   Hillcrest
   Pretoria
   0083
   South Africa
Editor's Notes

It is with sincere gratitude that SACJ takes leave of Dr Peter Lay who, until recently, was the assistant editor dealing with Information Systems. He has left academia for what sounds like a more gentle lifestyle. (He has gone farming!) Under Peter's stewardship the number of high-quality IS papers in SACJ grew steadily. In general, IS papers tend to be accessible and relevant to a wide spectrum of computer professionals, and the quality of IS papers that have been appearing in SACJ has significantly contributed to the increased interest being shown in the journal by the local computer industry. If this growth in interest is to be sustained, it is urgent and important to find a suitable replacement assistant editor. The ideal candidate should not only be respected as an academic by his peers, but should also be disposed to enthusiastically promote SACJ in the private sector. Since a shortlist of candidates is currently being compiled, I would like issue a general appeal for names that might be included on it. Please contact me urgently if you would like to be considered for the job, or if you would like to nominate someone that you consider to be particularly suitable.

My three year term of office as editor expires in October. I have always considered it a great privilege to hold this position, and as a result, I felt honoured when the SAICS executive committee requested that I stay on for a further term. Nevertheless, I initially declined the request on the grounds that the time-demands of the job were significantly eroding my ability to fulfil other duties. Particularly demanding has been the task of seeing to the typesetting of the various contributions - either by doing it myself, or by ensuring that it is adequately done by someone else. Recently, however, Prof G de V Smit (Riel Smit) at UCT has offered to assume the role of production editor. This generous offer so much changes the complexion of what is being asked of me that I am now both willing and honoured to continue as editor for another term. I am very grateful to Riel for his offer and I look forward to working with him. In future, authors whose papers have been accepted for publication will be asked to liaise directly with him regarding the precise form in which the final contribution should be submitted.

The next issue of SACJ will consist largely of a selection of papers that were presented at the 6th South African Computer symposium. The selection will be based on comments from the referees who, at the time, were asked to adjudicate the papers in terms of their appropriateness for both the conference as well as for SACJ publication. Papers which, in the opinion of one or more referees, required major revision will have to be resubmitted to SACJ for refereeing purposes. Authors will soon be contact in this regard.

At the time of writing, the updated list of "approved" publications for the first half of 1991 had not yet been released by the relevant authorities. For the sake of past, present and future contributors I sincerely hope that SACJ will be on the list when it eventually comes out. However, I have become increasingly aware that there is a real danger of laying too much store on papers published in so-called approved journals as a basis for evaluating and rewarding research. I hope to expand more fully on this theme in a future edition of SACJ. Keep watching this space!

Derrick Kourie
Editor

This SACJ issue is sponsored by a generous donation from UNIDATA
Abstract

The PEST System is a totally interactive, forward chaining, rule-based expert system shell. It was primarily, though not exclusively, designed for the development of scientifically related expert systems in Pascal, involving both symbolic and numeric computation. It accommodates the development of procedural knowledge, generates its own knowledge base code, embodies a simple but effective way of handling uncertainty, has a sophisticated interface, caters for different control mechanisms and contains an efficient explanation sub-system.

Keywords: Artificial Intelligence, Expert Systems, Production Systems

Computing Review Categories: 1.2

1. Introduction

Expert system shells facilitate the development of expert systems by providing a ready-to-run system which needs only the domain-specific parts to be added. They, typically, provide tools to ease the task of adding those parts in the form of rule editors and diagnostic aids. The design and implementation of the PEST System was an attempt to produce an expert system shell that incorporated the usual features of shells (explanation facilities, development aids, rule editor, etc.) together with the added features of parameterized condition and action procedures, access to the power of traditional programming language facilities, flexible control mechanisms and a fully interactive development environment, without any significant side effects.

In the development of a number of expert systems the need for parameterized condition and action procedures has been identified [2,4,7]. The use of parameters adds greatly to the power of a system. That a single condition/action routine can be employed over a wide range of situations is clearly of value in itself, but the more important point is that one can easily make parameter setting and alteration directly available to the domain expert without any need to go via the knowledge engineer.

The incorporation of algorithmic knowledge using a standard programming language has also been considered although, as Simons [7] points out, after a change in the knowledge base a compilation run is required, thus implying that such shells, although extremely fast and useful, are not well suited to a highly interactive environment. However, this compilation run even applies to the shells written in Prolog or Lisp where the rules are merely language clauses. The problem, seems to be more that the compiler, editor and shell are not a single unit, thus leading to user frustration. This problem can, therefore, be solved if it is possible to incorporate an incremental compilation stage for an existing algorithmic language into the interactive environment of the expert system shell.

2. Major Design Choices

The PEST shell was aimed largely, although not exclusively, at scientific domains where procedural knowledge would be allowed. It seemed clear, therefore, that these procedures should be written in a high level procedural language such as Pascal, and that these procedures should then be available to the particular expert system being designed. As has been pointed out by Waterman and Hayes-Roth [8] the lack of such a language for expressing procedural knowledge greatly slows down the development process and makes it difficult for the system to be extended by the users. However, to make the procedural knowledge more expert system oriented and to shield the end-user from most of the language dependent features, a fourth generation module was incorporated in the definition of rules. Previous attempts at the incorporation of procedural knowledge into the knowledge base have resulted in a poor interactive environment due to the separate development and consultation and to the "compilation run" required after every rule base change. By interfacing PEST directly with Borland International's Pascal command Line Compiler, it was possible to not only compile the procedural knowledge from within the shell, but also to interactively check and create code for a single rule on demand. A complete compilation of the shell was also never necessary since the majority of the code was pre-compiled and only linked in before consultation. The final expert system developed from the shell is a compiled ready to run program, completely separate from the development program. Shells with combined development and execution, such as VP-Expert, have limited available memory and are, as a result, a lot slower. The modular
design of Pascal enabled the generation and compilation of numerous modules from within the development program, and the linking of these and other pre-compiled modules, was then possible. A major problem in the implementation of expert systems on personal computers has been the lack of memory space available. In particular, large problems with many rules that cannot be all held in memory cannot be solved with these systems [2,5]. By using a paging system involving "dirty bits" and by using some of the features of Pascal, such as random-accessing of files it was possible to circumvent this memory problem.

The knowledge base structure of PEST was chosen to be rule-based as opposed to either a frame of a semantic net structure. Many systems have incorporated this structure and have shown that it works well in many domains. Dendral, R1 and Mycin are successful rule-based shells. Rules have the advantage that they employ a natural and easy style of problem solving, they allow for the easy incorporation of uncertainty and they can easily explain themselves. It was also decided that the rule selection method be forward chaining as opposed to backward chaining, not only due to its simplicity, speed and memory requirements, but because the choice was virtually forced by the incorporation of procedural knowledge. In a backward chaining system, the action part of the rule is checked to see if the current goal can be established. If the action part contains procedural knowledge, with possible parameters, it becomes impossible to test for goal establishment without placing severe restrictions on the procedural knowledge code or by a source code scan during consultation - a lengthy process.

Both the rule trace and canned text have been implemented as explanation mechanisms in PEST (see Figures 1, 2 and 3). The rule trace allows the user to page back through the most recently fired rules in the order in which they were fired. For each rule the conditions and actions of the rule, the confidence of each condition, the final confidence of the rule, and the canned text can be displayed. If additional strategic information is required, such as run-time information,
one of the actions in the rule can, of course, be an action that displays the required information.

A slight variation of the Mycin method [1] for calculating uncertainties within a rule was adopted. This choice was made, as opposed to the Dempster-Shafer of the Bayesian methods [3,6], not only due to the ease of implementation and user/expert comprehension, but also due to the fact that it accommodates the development of expert systems that both require and do not require uncertainty. The method used is as follows:

- The confidence factor of a string of facts combined by logical OR's is taken to be the maximum of the confidence factors of the individual facts.
- The condition confidence factor is then the minimum of the confidence factors separated by logical AND's.
- The conclusion confidence factor for the rule is the condition confidence factor multiplied by the confidence factor of the rule.
- To resolve possible conflict, if more than one rule has its condition confidence factor higher than the rule confidence factor, then the rule selected to be fired is the one with the highest conclusion confidence factor.

If the condition in a rule is a boolean test, then the confidence factor for that condition is either 0 (false) or 1 (true), depending on the outcome. If it is a condition function call, then the knowledge engineer is at liberty to allow the function to return a confidence factor in the range 0 (uncertain) to 1 (certain).

3. The Pest Development Module

The PEST system consists of three separate modules (see Figure 4), with each of the last two being generated and compiled by their predecessor. These modules are the Pest Development Module, the Expert System Module and the Consultation Module.

![Figure 4: Overview of the PEST modules](image)

The Pest Development Module is used by the knowledge engineer to develop the procedural knowledge base, the expert system base and to create the expert system module. In the Pest Development Module the knowledge engineer is shielded from the workings by statistic user interface. The components of the module are shown in Figure 5. The Procedural Knowledge Development component is used for the development and maintenance of all the procedural knowledge components. It includes an editor and a syntax checker. If any one component uses another, then that component is automatically checked first so that the most recent changes are made available to the using component.

Access is provided to the library of routines supplied with the shell. The editor makes these routines available by automatically inserting the relevant calls into the source. The Working Storage Base contains the Constants, Types and Variables defined by the knowledge engineer using standard Pascal declarations. The Tools Base is used to augment the tools already available within the shell. The knowledge engineer will thus code routines here that will be used in the condition and action routines. The Condition Base is simply a library of all the condition functions required by the expert system. The condition functions are all of type real. Each one will return a confidence between 0 and 1. The declaration of a condition function is:

```condition
FUNCTION ConditionName(parms): REAL;
```

The Action Base is a library of all the action procedures in the expert system. The declaration of an action procedure is:

```action
PROCEDURE ActionName(parms);
```

The Control Base contains routines that interact with the rule selection loop. This loop, which is automatically built into PEST, is of the form:
and the knowledge engineer is at liberty to include any code into the procedures; SessionInitialise, RepetativeRoutine, the boolean FinishedSession function, and the SessionTerminate routine.

The database development component allows the creation of up to 10 files each containing as many fields per record as required. Each field is, however, limited to one of the Pascal simple types. This database can then be developed and maintained from within the shell, tested, set or read within the procedural knowledge, tested within the conditions of any rule, and set within the actions of any rule.

Once the procedural knowledge and databases have been created, the module will automatically compile and link the defined modules into the Expert System Module.

4. The Expert System Module

The components of this module are shown in Figure 6. In this module the database can have data added to it or can have editing of values occur on records or fields. File names and fields are chosen via menus, and full input checking is performed. The rule base editor allows the entering, saving, checking, deleting and moving of rules. Editing assistance is offered here via pull down selection lists. This assistance is derived from the Pascal source entered by the knowledge engineer, from the database structure and from the current rule base state. Thus available conditions and actions may be selected from lists presented on the screen and added directly into the rule currently being handled. An example of the rule base editor display is shown in Figure 7. Rules are automatically checked for correct syntax before being saved and detected errors can be immediately corrected within the rule base editor. Each rule in the rule base consists of an alphanumeric name, a list of conditions, a list of actions, canned text and a rule confidence. The conditions are separated by the logical operators AND and OR. They can have one of the following forms:

- Conditions can be boolean tests, where the left of right operands are either function calls of data values (a constant, a working storage variable, a numerical calculation, or any legal Pascal combination of these). The operators can be any of the Pascal relational operators.
- A condition can be a condition function call from the conditions base. This function returns a real-value confidence in the range [0,1]. This function can be called with data value parameters if the argument is a value parameter and a working storage variable if the argument is a variable parameter.
- A condition can be a database test. This test is made via a boolean function call, passed with a database identifier, the record on which the test is to be made and a working storage variable against whose values the test is to be made. Such a call is of the form:

  \[ \text{DB_Test(FileName,FieldName,RecordNo,Variable)} \]

The consultation generation component uses information from the rule base verifier, the procedural knowledge (especially the control base), the data base, the rule base and other units to generate source code which it then compiles into the ready to run consultation module. Once this generation has occurred, the final system can be run from within the expert system module or can be run separately from disk.

5. The Consultation Module

The consultation module of PEST is the actual expert system that the end-user will finally run. As has already been stated, it is an executable file produced by other modules within the PEST system and which can be run independently of the rest of the system. The overall design of the consultation module is shown in Figure 8. The end-user is shielded from the inner workings of the system by both the static and the dynamic user interfaces. The static user interface is used when the system is explaining its actions, while the dynamic user interface is used during the execution of the consultation; mainly for entering data and displaying results. Initially the user is given the option of obtaining explanations during the consultations. After this the knowledge engineer’s SessionInitialise procedure is run followed by the main rule selection loop. After a rule is fired and if explanations are
The tool has achieved all of the original goals in providing a totally interactive system for designing non-trivial expert systems on personal computers in an efficient and powerful manner. The system is currently in successful use by students at the Honours level and is a proving a useful tool for both the teaching of expert system, and for research into the development of non-trivial expert systems on a personal computer.

References


Notes for Contributors

The prime purpose of the journal is to publish original research papers in the fields of Computer Science and Information Systems, as well as shorter technical research papers. However, non-refereed review and exploratory articles of interest to the journal's readers will be considered for publication under sections marked as Communications or Viewpoints. While English is the preferred language of the journal papers in Afrikaans will also be accepted. Typed manuscripts for review should be submitted in triplicate to the editor.

Form of Manuscript

Manuscripts for review should be prepared according to the following guidelines.

- Use double-space typing on one side only of A4 paper, and provide wide margins.
- The first page should include:
  - title (as brief as possible);
  - author's initials and surname;
  - author's affiliation and address;
  - an abstract of less than 200 words;
  - an appropriate keyword list;
  - a list of relevant Computing Review Categories.
- Tables and figures should be on separate sheets of A4 paper, and should be numbered and titled. Figures should be submitted as original line drawings, and not photocopies.
- Mathematical and other symbols may be either handwritten or typed. Greek letters and unusual symbols should be identified in the margin, if they are not clear in the text.
- References should be listed at the end of the text in alphabetic order of the (first) author's surname, and should be cited in the text in square brackets. References should thus take the following form:

Manuscripts accepted for publication should comply with the above guidelines, and may be provided in one of the following formats:

- in a typed form (i.e. suitable for scanning);
- as an ASCII file on diskette; or
- as a WordPerfect, TEX or LATEX or file; or
- in camera-ready format.

Charges

Charges per final page will be levied on papers accepted for publication. They will be scaled to reflect scanning, typesetting, reproduction and other costs. Currently, the minimum rate is R20-00 per final page for camera-ready contributions and the maximum is R100-00 per page for contributions in typed format.

These charges may be waived upon request of the author and at the discretion of the editor.

Proofs

Proofs of accepted papers will be sent to the author to ensure that typesetting is correct, and not for addition of new material or major amendments to the text. Corrected proofs should be returned to the production editor within three days.

Note that, in the case of camera-ready submissions, it is the author's responsibility to ensure that such submissions are error-free. However, the editor may recommend minor typesetting changes to be made before publication.

Letters and Communications

Letters to the editor are welcomed. They should be signed, and should be limited to about 500 words.

Announcements and communications of interest to the readership will be considered for publication in a separate section of the journal. Communications may also reflect minor research contributions. However, such communications will not be refereed and will not be deemed as fully-fledged publications for state subsidy purposes.

Book reviews

Contributions in this regard will be welcomed. Views and opinions expressed in such reviews should, however, be regarded as those of the reviewer alone.

Advertisement

Placement of advertisements at R1000-00 per full page per issue and R500-00 per half page per issue will be considered. These charges exclude specialized production costs which will be borne by the advertiser. Enquiries should be directed to the editor.
Contents

GUEST CONTRIBUTION

Why all the Fuss About Neural Networks?
G Barth ........................................................................................................................................ 4

RESEARCH ARTICLES

The Placement of Subprograms by an Automatic Programming System
J P du Plessis and H J Messerschmidt ............................................................................. 9

An Investigation into the Separation of the Application from its User Interface
J H Greyling and P R Warren ......................................................................................... 16

The Physical Correlates of Local Minima
L F A Wessels, E Barnard and E van Rooyen ............................................................... 22

An Efficient Primal Simplex Implementation for the Continuous 2-Matching Problem
T H C Smith, T W S Meyer and L Leenen ................................................................. 28

Concept Network Framework for a Multi-paradigm Knowledge Base
J Kambanis .................................................................................................................. 32

PEST - A Microcomputer Pascal Based Expert System Shell
A G Sartori-Angus and R Neville .................................................................................. 39

A Linda Solution to the Evolving Philosophers Problem
S E Hazelhurst .............................................................................................................. 44

TECHNICAL NOTE

Knowledge Representation using Formal Grammars
S H von Solms, E M Ehlers and D J Enslin ................................................................ 54

COMMUNICATIONS AND REPORTS

Book Review & Books Received ...................................................................................... 57
Cost-effective Visual Simulation Based on Graphics Workstations
A Caduri and D G Kourie ............................................................................................... 58
A Method of Controlling Quality of Application Software
T D Crossman .............................................................................................................. 70
An Update on UNINET-ZA: The Southern African Academic and Research Network
V Shaw ......................................................................................................................... 75