J M Bishop 20/3/90

South African Computer Journal
Number 1
January 1990

Suid-Afrikaanse Rekenaar-tydskrif
Nommer 1
Januarie 1990

# Computer Science and Information Systems

# Rekenaarwetenskap en Inligtingstelsels

**The South African Computer Journal**
*An official publication of the South African Computer Society and the South African Institute of Computer Scientists*

**Die Suid-Afrikaanse Rekenaartydskrif**
*'n Amptelike publikasie van die Suid-Afrikaanse Rekenaarvereniging en die Suid-Afrikaanse Instituut vir Rekenaarwetenskaplikes*

**Editor**
Professor Derrick G Kourie
Department of Computer Science
University of Pretoria
Hatfield 0083

**Assistant Editor: Information Systems**
Professor Peter Lay
Department of Accounting
University of Cape Town
Rondebosch 7700



**Subscriptions**

| | Annual | Single copy |
|---|---|---|
| Southern Africa: | R32-00 | R8-00 |
| Elsewhere: | $32-00 | $8-00 |

to be sent to:
*Computer Society of South Africa*
*Box 1714 Halfway House 1685*

# Editorial

At last the first edition of SACJ is available. I trust that readers will find it worth the waiting. There have been a number of teething problems in getting things together, the many details of which need not be spelt out here. One significant challenge was to cope with the consequences of the resignation of Quintin Gee, QI's highly competent production editor. He assisted in the initial phases of getting this publication together but had to resign for personal reasons. It is fitting to acknowledge here not only his initial advice and assistance in getting this first issue of SACJ off the ground, but also the many hours of work that he spent in previously producing QI.

Quintin's resignation meant that a new *modus operandi* for typesetting and printing had to be established. The exercise was not only time-consuming, but also has significant cost implications. Fortunately, the Unit for Software Engineering (USE) at Pretoria University has generously agreed to sponsor this first edition. On behalf of the South African computing community, I should like to thank them for their generosity. Now that they have made a first issue of SACJ possible, it is hoped to solicit the sponsorship of one of the larger computer companies for future editions.

It might be of interest to take readers on a walk through the new journal to highlight various aspects. To begin with, the cover design follows that of several journals whose titles have the format: *The South African Journal of Subject / Die Suid-Afrikaanse tydskrif vir Vakgebied* (where *Subject* and *Vakgebied* are appropriately instantiated). While colours vary, these journals generally have *Subject* and *Vakgebied* restated on the darker portion of the cover. SACJ's title was chosen in preference to a more descriptive but also more cumbersome title such as *The South African Journal of Computer Science and Information Systems*. The appearance of the words *Computer Science and Information Systems / Rekenaarwetenskap en Inligtingstelsels* on the cover are thus out of step with the original inspiration, but seem appropriate under the circumstances.

The inside cover is of interest for several reasons. Firstly, note that Peter Lay has kindly agreed to lighten my task by acting as an assistant editor. He will deal with matters relating to Information Systems. *Contributions in this area should henceforth please be sent directly to him.* Also note that an editorial board of distinguished persons has been assembled. I should like to once again thank board members for adding status to SACJ by agreeing to serve in this capacity. They will be consulted on matters of editorial policy whenever appropriate. Finally, the subscription costs have been increased to keep pace with production costs. This increase does not affect SAICS members, who will continue to receive the journal as one of the benefits of membership.

The guest editorial by Pieter Kritzinger makes for interesting reading. Several points of concern about computer-related research in South Africa are raised. I trust that the article will focus attention on these problems and stimulate a debate which will lead to eventual solutions. It is hoped to make guest editorials a regular feature of future SACJ issues.

Of the eight research papers offered in the journal, four have been gone through the normal channel of refereeing and revisions. The remainder were submitted to the Vth SA Computer Symposium and are published here by invitation. Each paper submitted to the chairman of the symposium's program committee was sent to three referees. A ranking scheme, reflecting an aggregate measure of referee evaluation, was used as a basis for deciding on papers to be presented. After further editorial evaluation, the authors of four of the five highest ranking papers were invited to submit their papers to SACJ. While it was not possible to contact the fifth author in time for this edition, but it may be possible to publish that paper, together with a selection of others from the symposium, in future SACJ editions.

In the section marked *Communications* various items of news arriving at the editor's desk have been published. It was particularly gratifying to receive book review submissions in response to a prior general appeal. There has also been an enthusiastic response from book publishers, who have sent in a number of books for review. Titles are listed in the *Communications* section. Please contact me if you are willing to review one (or more) of these. Naturally, reviews of other books of interest in your possession will also be welcomed.

The final point to highlight in this walk through the journal is the increase in page charges indicated on the back inside cover. These reflect the increased cost of production. Since research papers in SACJ qualify for state subsidy at academic institutions, the charges should not, in general, present major problems for authors. However, it is worth pointing out that the final format of papers submitted significantly impacts on both the financial and editorial load. Submissions in camera-ready format (or nearly so) result in both a cost savings and a speed up of turn-around time by several orders of magnitude. Since many readers may not be familiar with the printing process, it may be helpful to say something about it in order to substantiate this claim.

The printing process basically involves typesetting, shooting (or photographing), and then reproduction and binding. Apart from limiting the amount of material, the printer's client has very little control over the cost of shooting, reproduction and binding. On the

other hand, anyone equipped with moderate text- or word processing facilities and a laser printer can go a long way (if not all the way) towards typesetting a paper. Even a partially typeset paper helps significantly, as I will explain below.

By typesetting I simply mean knocking the paper into the right shape and producing a laser printout. The printers regard this is a tedious, error-prone task, even if they start off with an ASCII file rather than a hardcopy of the paper. Consequently, they tend to handle large-scale typesetting by subcontracting the task. Moreover, while they may be willing to typeset uncomplicated text, they tend to balk at text containing specialized mathematical and other notation. However, they are quite skilful at cutting and pasting text, and at enlarging or reducing photographed or scanned diagrams. They are even willing to redraw sketches which are not too complicated.

As a result of the above, I have pressed several authors to do their own typesetting. In cases where it was problematic to produce double column format, a single column of appropriate width was requested. While this is a second-best option, it allows for cutting and pasting to be done by the printers. Some sketches have either been directly reduced from the author's original, while others have been redrawn by the printer. By way of exception, I have personally undertaken the typesetting of a few papers using WordPerfect. However, I would like to avoid this as far as possible in future, and consequently appeal to potential authors to make every effort to do their own typesetting.

From SACJ's point of view encouraging authors to do their own typesetting involves a compromise in that there will inevitably be slight variations in the print from one article to the next (as is in fact the case in this issue). If you are pedantically inclined, you might consider this to be a disaster. Personally, I regard it as a rather neat advertisement for the typesetting skills of SACJ contributors.

As an aside, since the handling of TEX files was initially a problem for me, I was pleased to discover that Peter Wood and his colleagues at UCT have mastered the art of producing TEX printout in the format now before you. Future authors who use TEX should consult them on details.

As to the future, it is not possible at the this stage to commit to a fixed number of SACJ issues per year. The number of issues is constrained by finance, submissions of the right quality, and time available to the editorial staff (including our anonymous and unsung heroes - the referees). The ideal is to produce four issues per year, but this may not always be attainable.

In conclusion, if readers have as much fun in reading this first issue of SACJ as I have had in editing it, the hours spent on it will have been well worthwhile. Hopefully SACJ is destined not only to be a permanent feature of the Southern African computing scene, but also to significantly contribute to research in the region.

**Derrick Kourie**
**Editor**

This SACJ issue is sponsored by
**The Unit for Software Engineering**
**(USE)**
Department of Computer Science
Pretoria University

# Guest Editorial

## Funding Computer Science Research in South Africa

P S Kritzinger
*Department of Computer Science, University of Cape Town, Rondebosch 7700*

The word *research* has many connotations and is often abused. In everyday language a person does not simply *search for information in a library*, for example, but rather does *research*, thus pretentiously conferring an aura of intellectual activity on an effort which requires very little original thought.

Here I will interpret the term to mean work which generates results that gain international recognition. This implies that the work is published in good international journals or presented at international conferences. I believe this is the only valid index of the quality of research.

With very few exceptions, the computer industry in South Africa is a consumer of computer technology, rather than a developer. In contrast with, say, the chemical industry, there is therefore no tradition of research in computer science in the South African computer industry and computer science researchers therefore have, as virtually their only source of funding, the Foundation for Research Development (FRD) which has its origins in the CSIR.

The FRD was formed in April 1984 with the development and use of research expertise in the natural and applied sciences and engineering as its mission. This mission is primarily directed at the universities, museums and technicons with the ultimate aim of improving the life of all South Africans.

Although the FRD has several programmes, the two which are of main concern to computer scientists are the Core Programmes and the Special Programmes.

FRD Core Programmes foster the optimum development of a scientific and technological knowledge base by supporting individual self-initiated research. These programmes, started only about 4 years ago, have met with considerable acclaim, particularly in regard to the way in which research funding for a particular individual is decided. To qualify for support within a Core Programme, researchers must obtain a certain evaluation status within the FRD and funding is then linked directly and exponentially to the merit of the individual concerned, rather than being linked to the specific project proposed.

In the evaluation process, peer review is strongly emphasised. The researcher himself is expected to nominate referees, whose status and reports play a decisive role in the evaluation. As a result of this evaluation, an applicant is assigned a specific evaluation status category. There are currently 9 categories in all, but the ones of main interest are:

- **A** researchers who are without any doubt accepted by the international community as being amongst the leaders in their field (52);
- **B** researchers not in category **A** but who nevertheless enjoy considerable international recognition as independent researchers of high quality (182);
- **C** proven researchers who have maintained a constant high level of research productivity and whose work is regularly made known internationally, or proven researchers whose current research output is less but who are actively engaged in scholastic activity (433);
- **P** researchers younger than 35 years of age who have already obtained a doctoral degree and who have shown exceptional potential as researchers (10); and
- **Y** young researchers usually under 35 years of age, who are highly likely to achieve C status by the end of their support period (108).

The number of researchers in the various categories as of August 1989 has been indicated in parentheses above. Of these, only 7 persons are computer scientists: 1 in category B; 3 in category C; and 3 in category Y. Only 4 departments of computer science are involved.

The other main programmes of concern to computer persons are the Special Programmes which aim at developing research manpower in priority areas. After identification of an area that merits particular research development, given local expertise, a Special Programme is launched to address the problem in the national interest.

Although a manager of a Special Programme has to be an FRD evaluated researcher, the same need not be true for the other team members. Regular peer evaluation of researchers as well as evaluation of the progress and results of Special Programmes are considered essential. Special Programme awards will be made for the first time towards the end of 1989. It is therefore not yet known whether proposals already submitted for programmes in computer science have been successful.

It is clear that, in the context explained above, there is virtually no computer science research being done in South Africa - a scary thought which has considerable implications for this country! Why is this so? There are several reasons, but I would like to single out two in particular.

Qualified faculty and students is an abiding problem at the heart of computer science departments. Acquisition of new faculty members is an issue intimately linked to the number of graduate students successfully completing PhD degrees. This problem is by no means unique to South Africa. For instance, data gathered in North America indicates that in 1983 there were over 200 vacancies in the 91 departments that have doctoral programmes in computer science. At the same time, only approximately 250 PhD's were granted in North America - a figure that has remained relatively unchanged for the past several years. A large number of those graduates were attracted to industry and industrial research laboratories. Although I do not have solid data at my disposal, I would think that South Africa produces at most one PhD graduate in computer science per year. There are currently 20 departments of computer science at universities is South Africa. It will therefore take us 20 years to locally produce one new faculty member with a PhD in computer science for every university.

Contributing to the above problem is our current academic image. The graduate student usually sees concerned computer science faculty members as rather harried individuals, having large undergraduate classes, much committee and professional work, and labouring under an ill-fitting model (applicable to more established disciplines) for decisions on tenure, salary and promotion. Further, as undergraduates, many prospective graduate students were not engaged in research projects involving computer science faculty, and for that reason were not exposed to graduate students doing research, and rarely developed a camaraderie with any computer science professionals. At last count there were only 5 individuals in South Africa who completed their computer science doctorate at a university outside South Africa where they had the good fortune to work in an environment in which sufficient faculty and funds were available to create an ethos of research. It is difficult to convince students that their interests and goals can be served by a PhD in computer science or by an academic career.

The second problem, which is of greater concern to me since there is no immediate solution to it, has to do with the fact that senior persons who decide the fate and fortune of academic computer science departments are, in general, individuals whose professional careers started well before computing machines came into every day use - that is to say, in the years B.C. (Before Computers). These persons of influence do not always understand what "computers" are, and what their potential influence upon the workplace in particular and society in general are. As far as research (as opposed to teaching) is concerned, most of them understand that a medical school needs special and expensive equipment (not to mention, expensive faculty) and that engineers must have a workshop and special machinery to teach their students and conduct research. They understand that if one needs to build up a defense industry, it will cost billions of rands; but they are not so sure about computer science, even though many other countries have recognised it as of national strategic importance.

I believe that only time and dedication will lead to a solution of these seemingly insurmountable problems and allow computer scientists to take their rightful place in the research community in South Africa.

## Bibliography


P J Denning, [1981], Eating our seed corn, *Communications of the A.C.M.*, **24**(6), pp.341 - 343.

J Tartar (Ed.), [1985], The 1984 Snowbird Report: Future issues in computer science, *Communications of the ACM*, **28**(5), pp490 - 493.

D Gries, R Miller, R Ritchie and P Young, [1986], Imbalance between growth and funding in academic computing science: two trends colliding, *Communications of the A.C.M.*, **29**(9), pp.870 - 878.

J E Hopcroft and D B Krafft, [1987], Towards better computer science, *IEEE Spectrum*, pp.58 - 60.

# The Use of a Lattice for Fast Pattern Matching

G. Deon Oosthuizen
*Department of Computer Science, University of Pretoria, 0083 Hillcrest*

## Abstract

*Pattern recognition involves the matching of a given pattern against a collection of patterns in store in order to identify the given pattern as belonging to a particular class already known to the system or not. Similarly, data retrieval from content addressable storage involves the matching of a partial pattern against a collection of patterns in store in order to retrieve the required pattern.*

*This matching procedure can be made very efficient by integrating stored patterns into a graphical structure called a lattice. The lattice forms an optimal indexing mechanism and supports fast parallel matching. The indexing mechanism is optimal because it is both complete and minimal: all possible indices to a particular pattern are generated, and indices are integrated maximally, i.e. paths to similar patterns overlap as much as possible and thereby prevent duplication. Moreover, the indexing mechanism is robust: if faulty or partial information is supplied, the lattice is guaranteed to provide the best approximate answer.*

***Keywords:*** *Knowledge Processing, Matching, Clustering, Indexing, Information Retrieval*
***CR categories:*** *I.2.4, I.5.1, H.3.1*

Presented at V th S.A. Computer Symposium.

## Introduction

Knowledge processing involves the handling of discrete symbolic entities. These entities are often complex conglomerate structures, e.g. *frames*, expressed in terms of strings of features.

Inevitably, the processing of these complex entities involves matching them against one another. *Retrieval* requires matching: partial information regarding an entity is supplied for the purpose of extracting all the stored information concerning the entity or a set of entities. *Updating* requires the same kind of matching, but with the purpose of updating the information concerned. Even *inference* can be regarded as a kind of pattern completion procedure and requires matching. This is well illustrated by the operation of rule--based systems. During the operation of such systems, each rule in a rule base is matched against a large set of facts stored in memory. Equating each rule to a database query, rule base execution can be regarded as the simultaneous execution of many queries - as many as there are rules - over a slowly changing modest size database [1].

In this paper we describe how pattern recognition can be performed very efficiently by storing patterns in a structure called a lattice. The method represents a connectionist approach to pattern matching employing a novel network structure. The only related work is the NETL system developed by Fahlman [2], to which we refer again later.

We first present an outline of an algorithm developed to construct the lattice, followed by a simple example illustrating it. We then discuss the inherent clustering capabilities of the lattice as well as the relation between clustering, indexing and matching. We explain how the lattice is used as a fast parallel matching mechanism. We conclude by referring to other properties of the lattice that are of interest to knowledge processing.

## Construction of the Lattice

The lattice is constructed by starting off with an 'empty' graph and adding the patterns to be stored one by one. An algorithm has been developed [3] for performing the necessary transformations. The algorithm accepts vectors (patterns) of discrete--

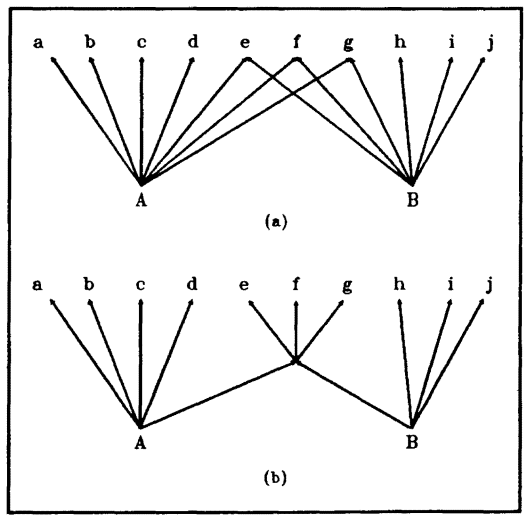


**Figure 1**

valued attributes (features) and merges them into a network structure. At first a node is created for each value of each attribute to be recorded. These nodes can be considered as the upper layer of a network to be expanded below them. For the first pattern to be stored, a single node is created below the initial layer and connected to each of the features (in the initial layer) which is present in the pattern. Thus, the pattern is represented by a node linked to each of the feature elements characterising the pattern.

The same is done for the second pattern to be stored, with the difference that if the second pattern has any elements in common with the first pattern, a third node is created and connected to all the original feature-nodes which the first two patterns have in common (see fig. 1). The nodes representing the first two patterns are then connected to the third node instead of the feature-nodes which they have in common - or these links are deleted where they already existed.

This procedure is repeated for every subsequent pattern to be stored, making no distinction between result of reconfiguration - referred to as intermediate nodes. Before a new node is created for a pattern volunteered, the pattern is matched against existing nodes to determine if it is already captured by existing nodes. If this is the case, the *confidence factor* (explained below) of the existing node, as well as all nodes above it in the network, is raised by one.

The resultant network configuration constitutes a *lattice* (technically speaking, a *sublattice*) and is unique for a given set of patterns. Every pair of elements in a lattice has a unique least common superior (the *join*) and a unique greatest common subordinate (the *meet*). In other words, among the common subordinates of any two nodes there is a unique node which is above (greater than) the others. Similarly, there is a unique lowest, or least, common superior. In lattice theory, these are referred to as least upper bounds and greatest lower bounds respectively. To construct a proper lattice, the feature-nodes have to be connected to a single common superior, and the pattern-nodes to a common subordinate. We omit them. (Since the focus in this paper is on the application of the lattice rather than its construction, the ideas are discussed with reference to informal examples only. A formal description of the algorithm can be found in [3].)

Thus, each subset of features has a unique meet (if it does have a meet) and each group of patterns has a unique join (if it does have a join). Conversely, each intermediate node has an associated group of features and an associated group of patterns. It is said to *span* the nodes above it and *cover* the nodes below it. We refer to the upper nodes in the graph (the feature-nodes) as *classes* and the lower nodes (the patterns) as *objects*. The nodes created in between are called intermediate nodes as we mentioned before. The intermediate node (or object) that forms the root of a tree of which a number of classes form the leaves, constitutes their mutual meet. The confidence factor of a node is an integer value stored with the node indicating of how many patterns contained the set of features (subpattern) denoted by the node.

**Example**

Let us consider the following example taken from Oosthuizen [3]. A group of 9 individuals (see Table 1) are classified on the basis of 5 features each. The attributes are size, hair colour, eye colour and complexion. The classification is fictitious and can have any meaning. For our current purposes we view each individual as described in terms of a pattern of features of length 5, i.e. the class is considered as another feature. Fig. 2 shows some of the 26 intermediate nodes (indicated by '*'-nodes) created during lattice construction. Notice that this is only a partial graph. Some arcs and object S8 have been deliberately omitted for the sake of simplicity.

| Example no. | Length | Hair | Eyes | Complexion | Class |
|---|---|---|---|---|---|
| S1 | short | blond | blue | fair | positive |
| S2 | tall | blond | brown | dark | negative |
| S3 | tall | red | blue | fair | positive |
| S4 | short | red | blue | dark | negative |
| S5 | tall | grey | blue | fair | negative |
| S6 | tall | blond | blue | fair | positive |
| S7 | tall | grey | brown | dark | negative |
| S8 | short | blond | brown | dark | negative |
| S9 | tall | blond | brown | fair | negative |

Table 1

## Clustering

The transformation algorithm has the effect of capturing all similarities between patterns in a series of "tangled" hierarchies between the feature-nodes at the top and the pattern-nodes at the bottom. Each path from the bottom to the top of a hierarchy contains a sequence of nodes, covering larger and larger sets of patterns. Consider the sequence

S1, *10, *1, *6, BLUE.

S1 is a pattern. *10 identifies the smallest cluster of patterns associated with S1 - i.e. those patterns most similar to S1. The patterns in this cluster, S1 and S6 are very similar and differ only in respect of the value of the LENGTH attribute. As we progress upward, we obtain larger clusters containing patterns which have less in common.

Each node has an associated set of features (those spanned by the node) which serves as description of the cluster of patterns denoted by the node (those covered by the node). E.g. the description for node *6 would be: blue **and** fair.

These features are the ones which the patterns in the cluster have in common. All the patterns covered by *6 have the above two features in common. The higher up in the graph the node is, the shorter is its associated subpattern of features, and the larger the cluster of patterns it denotes. Thus we obtain an implicit clustering mechanism grouping similar patterns together in clusters of increasing size.

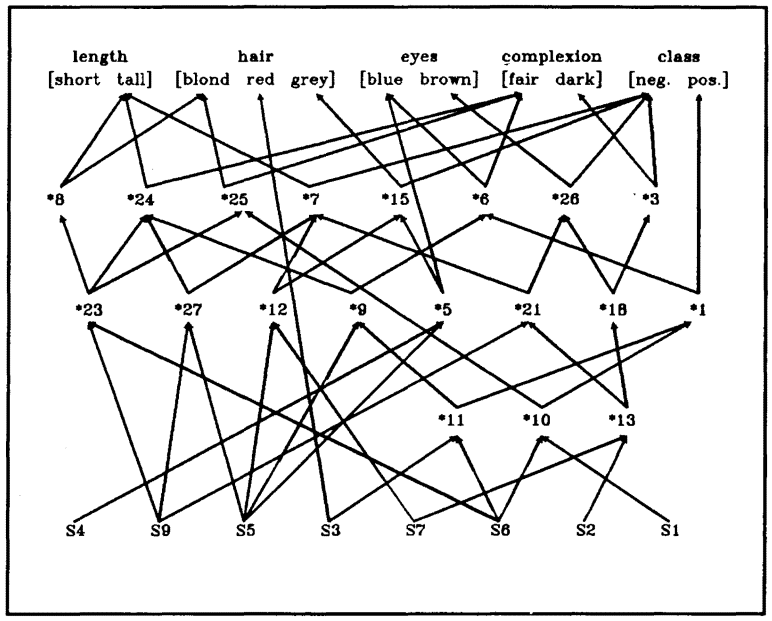


**Figure 2**

Each pattern is characterised by the features above it in the graph. Each pattern finds itself at the bottom of a number of hierarchies, each denoting a cluster of patterns of which the pattern in question is a member.

## Clustering and Indexing

The goal in fast data retrieval is to be able to retrieve an arbitrary collection of data from mass data storage. Since the early days of the creation of large direct access files on secondary storage, suitable indexing mechanisms were constructed to enable the system to find a quick path to the right data. The same principle applies to the use of inverted files in database systems: the objective is to rapidly identify a suitable subset of the total set of data items for further processing. This is done by storing references to that which is similar in a particular respect, together in a separate index. In that sense, indexing is closely related to clustering. The only difference is that in these systems, the emphasis is on ordering, i.e. clustering is obtained by ordering data and retrieval is achieved by identifying an appropriate segment in an ordered range.

In content addressable systems, retrieval essentially becomes a matching task and the emphasis shifts from ordering to clustering - clustering in such a way that coherent subsets of the data can be accessed in an arbitrary way. The key aspect involved, is the ability to retrieve any subset (cluster) of objects by specifying an arbitrary list of features. It is in this respect that the lattice presents an ideal solution.

The lattice provides a complete mapping between classes and objects. Consider the following view of the lattice:
The lattice contains a node for (and only for) each instance where a member of the set of all possible clusters of classes corresponds to a member of the set of all possible clusters of objects. In other words, for every possible cluster of objects (patterns) which could be identified by a unique set of features, there exists a unique node in the graph which relates them to one another. Conversely, for every combination of features which is substantiated by one or more objects, there exists a node (their meet) which serves as direct link to the relevant object(s).

We conclude therefore that the graph constitutes an ideal indexing mechanism. The lattice forms an indexing mechanism which is optimal because it is both complete and minimal: all possible indices to a particular pattern are generated, and indices are integrated maximally (as we explained earlier), i.e. paths to similar patterns overlap as much as possible and thereby eliminate duplication. In the next section we explain how it is used.

## Matching

(a) *Retrieval:*
If a pattern is volunteered to the system, it is determined if its corresponding features have a meet in the graph. If they do have a meet X, say, two conditions may hold. X may be an object, in which case a single pattern is matched and retrieved. If X is an intermediate node, a group of patterns is retrieved, i.e. it means that the given pattern is only a partial pattern which several stored patterns have in common.

If the matching mechanism is used and no pattern fits the specified description, the matching procedure delivers the pattern or cluster of patterns which is most similar to the specified pattern, i.e. it delivers the best approximation. When this happens, the features of the input pattern have no meet in the graph. The meet of the largest subset of the input features which *do* have a meet, is then lo-

cated. As in the above case of an exact match, the meet delivered by a partial or approximate match can be an object - denoting an individual pattern - or an intermediate node, denoting a cluster of patterns.

If two or more equally large subsets of features with meets are encountered, the meet with the highest confidence factor is selected. The meet with the highest confidence factor denotes the subpattern which occurred most in the past. This subpattern is regarded as dominant. The application of this heuristic is equivalent to assuming the best *default* values - determined on the basis of all patterns encountered up to the current point in time.

To understand why lattice-based matching provides the best approximation, the matching procedure can be viewed as follows. The determination of the meet literally implies intersecting the clusters corresponding to each of the features in the specified pattern in an attempt to determine if they overlap. Approximate matching takes place if not *all* of them overlap but *some* of them do overlap. The largest overlapping group delivers a pattern or a cluster which fits the input pattern best, because it has the largest number of features in common with the input pattern.

For example, let us say we want to retrieve from the system shown in fig. 2 the following pattern:

TALL, RED, BROWN, DARK, NEGATIVE. (1)

Each of the features in this pattern has a corresponding node in the network that covers a cluster of patterns: the set of stored patterns having the particular feature in common. The meet of a number of feature-nodes covers a cluster which constitutes the intersection of these clusters. In the case of the above pattern, however, the features do not have a meet (see fig. 2). This means that the specified clusters do not intersect and that no pattern containing the above set of features has been stored yet. Still, the clusters covered by

TALL, RED (2)

intersect, namely at node S3, and so does

TALL, BROWN, DARK, NEGATIVE (3)

namely at S2 and S7 - a cluster represented by *13, the meet of these features. Node *13 denotes the intersection of four clusters whereas S3 denotes the intersection of just two. The latter subpattern (3) has more in common with the specified pattern (1) than does the former one (2). Thus, (3) is the best approximate match for the specified pattern and returns a cluster of patterns denoted by *13.

If retrieval of a single pattern is a requirement, or if there has to be 'homed in' on a smaller subset of a cluster retrieved, the procedure described below applies.

(b) *Classification/Recognition*:

In each of the cases discussed above the matching procedure either delivers an individual pattern or it delivers a cluster of patterns. If the goal is recognition, the discovery of a single matching stored pattern means that the input pattern can be uniquely identified, and signifies the end of the operation. If the meet denotes a cluster, then a single member of the cluster still has to be selected. It means that, based on the information supplied, the best the system could do was to identify a cluster of patterns of high similarity with the input pattern. It can be regarded as a preliminary step during which the best candidates for the final match is selected. The next step would be to test each of the candidates against the environment, i.e. the features present in the candidates but not present in the input pattern have to be tested against the environment to determine which pattern is preferable. This has to be done in such a way that as many as possible candidates are eliminated with each test.

Once again the properties of the lattice comes to bear. Let us say the cluster retrieved is denoted by an intermediate node X. The subordinate nodes of X will denote either individual patterns or subclusters of X, each containing at least one feature in its description which is not in the description of X, i.e. which distinguishes it from X. By conducting a breadth first search from X downward, we can ensure that the most discriminatory features are considered first.

For example, let us say the input pattern consists of the following (see fig. 2):

TALL, BLUE, FAIR, BLACK.

Let us assume that the hair colour BLACK is recorded but does not occur in the stored patterns. As a result the above features will not have a common meet. However, three of them (TALL,BLUE,FAIR) do have a meet, *9, which will be located upon matching. Now, *9 has two subordinates, S5 and *11. The description of S5 differs from that of *9 by containing the additional features GREY and NEGATIVE. Similarly the description of *11 contains POSITIVE additionally. These are the features which have to be tested for compatibility with the environment, e.g. a test may be conducted to determine the true class: NEGATIVE or POSITIVE. The additional information is then used to resolve the issue, i.e. to select the best candidate.

One aspect which is not well illustrated by the example is the fact that in complex domains it is very useful, if not essential, to be led by the system in considering additional features. For example, in document retrieval it may happen that the initial pattern of keywords supplied by a user may cause the retrieval of a very large set (cluster) of documents. There could be a large number of potential keywords which the user could supply additionally in order to prune this set and it is often not known which keywords are applicable. The lattice would enable the system to guide the user by suggesting suitable keywords to reduce the size of the goal set.

When the user does not know how to focus the search, the system will guide him by prompting him, starting with the most general terms (keywords). The lattice imposes a partial ordering on the clusters of patterns which enables the user to navigate through the possible sets in a meaningful way. (Note that in the same way as we have described how smaller clusters can be considered, larger clusters can be considered as well.)

## Implementation

Two operations have been defined on the lattice. The upward closure of a node includes the node itself and all the nodes above it in the graph. The downward closure includes the node and all the nodes below it in the graph. The meet of a number of classes is determined by computing the intersection of the downward closures of the classes involved. The closure operations can be implemented by means of parallel marker propagation.

Fahlman [2] employs parallel marker propagation for a similar purpose in a system called NETL, referring to it as a "fast-intersecting network". However, Fahlman's network is not a lattice and therefore it does not have the properties of a lattice referred to above. It is a normal semantic network which contains no non-lexical nodes (nodes without symbolic labels) as the lattice does in the form of the intermediate nodes.

The transformation algorithm is of polynomial complexity (the computation time increases polynomially with the number of features per pattern). The weakest point of the method is that the number of intermediate nodes is liable to explode if the features recorded are independent or if the patterns contain noise (incorrect data). To counteract this, a pruning mechanism has been devised for removing insignificant nodes from the graph [3].

A serial implementation on a Sun-3/180 computer takes less than a second to match patterns of length ten with a network containing approximately 600 intermediate nodes and 270 patterns. The target hardware, a Generic Associative Array Processor [4], typically speeds up closure times by a factor 10.

## Other Properties of the Lattice

We mentioned that inference can be viewed as a kind of pattern completion operation. The application of the above matching capabilities allows the lattice to function as a rule base, to be specific, as an *optimally integrated* rule base: a rule base in which each term maximally appears once in the rule base as a whole, although it may form part of several rules [5]. Moreover, the rules involved are induced from examples. The lattice forms the basis of a conceptual clustering mechanism. Each of the clusters referred to earlier on constitutes a *generalisation* used by the system to form concept descriptions.

## Conclusion

In this paper we explained that a lattice constitutes an ideal indexing mechanism which facilitates fast pattern matching for knowledge processing. Additionally, in the event of contradicting information when only a partial match is possible, the system is guaranteed to perform the best approximate match by automatically substituting conflicting or missing features with most likely default values or by suggesting appropriate substitutive values for evaluation. The strength of the technique lies in the combination of parallelism - afforded by the connectionist nature of the architecture - with the inherent properties of completeness and optimality of the lattice.

## References


[1] D P Miranker, TREAT: A Better Match Algorithm for AI Production Systems. *Procs. AAAI-87: Sixth National Conference on Artificial Intelligence*, 1987.

[2] S E Fahlman, NETL: A system for representing and using real world knowledge. The MIT Press, Cambridge, MA, 1979.

[3] G D Oosthuizen, The use of a lattice in knowledge processing. *PhD Thesis*, University of Strathclyde, Glasgow, 1988.

[4] D R McGregor, S T McInnes, M Henning, An aarchitecture for assciative processing of large knowledge bases. *The Computer Journal*, vol. 30 (1987), no. 5, pp. 404-412.

[5] G D Oosthuizen, D R McGregor, Induction through knowledge base normalisation. *Procs. of European Conference on Artificial Intelligence*, Munich. Pitman Publishing, 1988.



**Acknowledgements**

The lattice transformation algorithm was developed while I held a research position in the Department of Computer Science at the University of Strathclyde, Glasgow. The work was supported by the British Department of Trade and Industry. I am indebted to my former colleagues, and to Jesus Christ, the Origin of knowledge.

## Computers and the Law

*Submitted by Antony Cooper*
*CSIR*

The SA Law Commission has established a commission on "The Legal Protection of Information".

The commission is still in its preliminary stages and the assigned researcher, Mr Herman Smuts, is still preparing the working paper. He does not know when it will be finished, but once the working paper has been prepared, they will invite comments for about two years, before preparing the final report. I have contacted Mr Smuts, and he would be most grateful to receive input at this stage, especially regarding the terms of reference of the commission. His address is:

C/o SA Law Commission
Private Bag X668
**PRETORIA**
0001

In addition, there is an ad-hoc committee at the Registrar of Copyright investigating numerous copyright issues, including those relating to software and data. Mr Smuts' commission will be liaising with the ad-hoc committee.

I feel that SAICS has an obligation to submit evidence to the commission, and I would appreciate it if you would circulate the members of the Council of SAICS, and perhaps the general membership as well, to solicit ideas concerning SAICS's input.

I shall prepare something for the commission, either in my personal capacity, or in my professional capacity here at CSIR. I would be willing to assist in the preparation of any evidence SAICS might submit.

## 4th National MSc/Phd Computer Science Conference

*Report by Danie Behr*
*University of Pretoria*

This conference was held from 7th to 10th September 1989 at the Cathedral Peak Hotel in the Drakensberg. The conference was attended by 61 postgraduate students from 11 South African universities. Most were engaged in MSc studies, although 5 Phd students also attended. These numbers are encouraging for the South African computer science community. This type of conference is rather unique in that it affords students the opportunity of sharing their research, and getting to know other researchers in the country. The number of Afrikaans and English speaking students attending the conference were roughly equal. Presentations were made in the language preferred by the student. Invitations were sent to all universities with computer science departments. The conference was organized by the students themselves.

Some of the more popular research topics that were presented included expert systems, data communications, computer security, graphics, software engineering, user interfaces and data bases. The main sponsor for this year's conference was the Division for Microelectronic Systems and Communication Technology of the CSIR. The conference was opened with an interesting talk on the myths and motivations of post graduate studies by Prof DG Kourie, acting head of the Computer Science Department at Pretoria University.

The next conference will be presented by the University of Port Elizabeth. People requiring further information about the next conference should contact Andre Calitz, Charmaine du Plessis or Jean Greyling of the Department of Computer Science at UPE.

A list of authors and papers presented at the symposium follows:

S Crosby, University of Stellenbosch
*Performance Analysis of Wide Area Computer Communication Networks*

A B Joubert, PU for CHE Vaal Triangle Campus
*Image Processing Libraries*

A Calitz, University of Port Elizabeth
*An Expert System Toolbox to assist in the classification of objects*

L von Backström, University of Pretoria
*Integrated Network Management*

R Foss, Rhodes University
*The Rhodes Computer Music Network*

A McGee, University of Natal
*On Fixpoints and Nondeterminism in the Sigma-Lambda Calculus*

P G Mulder, Randse Afrikaanse Universiteit
*A Formal Language and Automota approach to Data Communications*

A Tew, Randse Afrikaanse Universiteit
*Drie dimensionele grafiek grammatikas*

T C Parker-Nance, University of Port Elizabeth
*Human-Computer Interaction: What Determines Computer Acceptance*

E Coetzee, PU vir CHO Vaaldriehoekkampus
*Opsporing van rande in syferbeelde dmv verskerping en drempelbepaling*
D A Sewry, Rhodes University
*Visual Programming*
A Cooper, University of Pretoria
*Improvements to the National Exchange Standard*
E S Badier, University of Port Elizabeth
*A Computer Assisted Diagnostic System (CADS)*
C du Plessis, Universiteit van Port Elizabeth
*Persoonsidentifikasie dmv naampassing in 'n genealogiese databasis*
J Greeff, University of Stellenbosch
*The Entity-Relationship Model and its Implementation*
D A de Waal, PU vir CHO
*Flat Concurrent Prolog (FCP) en Flat Guarded Horn Clauses (FGHC): 'n Vergelyking*
E Naude, UNISA
*Interne metodes in Liniêre Programmering*
A Deacon, University of Stellenbosch
*Global consistency in non-locking DDBMS*
A Wilks, Rhodes University
*The Synchronisation and Remote Configuration of the Resources in a Computer Music Network*
J Greyling, University of Port Elizabeth
*The design of a User Interface with special reference to an Interactive Molecular Modelling Program*
L Drevin, PU vir CHO
*Rekenaarsekuriteit: Verskillende vlakke van kontrole*
Dieter C Barnard, University of Stellenbosch
*The design and implementation of a modest, interactive proof checker*
R A Schmidt, University of Cape Town
*Knowledge Representation Systems and the Algebra of Relations*
J Hartman, Randse Afrikaanse Universiteit
*Die Gebruik van Objek-georiënteerde Programmering in die Moderne Sneltrein Omgewing*
S Lawrie, Rhodes University
*The Design and Implementation of a System for the Interactive Control of a MIDI-based Studio*
E Mulder, Rand Afrikaans University
*A Formalisation of Object-Oriented Principles*
C J Tolmie, UOVS
*Die Ontwikkeling van 'n Ekspertrekenaarstelsel vir die beoordeling van die resultate van die Technicon H1-Bloedselanaliseerder*
R Breedt, University of Pretoria
*Realism with Ray Tracing*
J van Jaarsveld, University of Pretoria
*Developing Medical Expert Systems: A knowledge acquisition perspective*
W Appel, University of Pretoria
*TCP/IP Implementation on Ethernet*
E Goedeke, University of Natal
*Eggspert's Control Structure*
M Harmse, University of Stellenbosch
*Modelling of I/O Subsystems*
H L Viktor, University of Stellenbosch
*A Quantitative Model for Comparing Recovery Techniques in a Distributed Database*
M Olivier, Randse Afrikaanse Universiteit
*Rekenaarvirusse in Suid-Afrika*

# Book Reviews

**An Introduction to Functional Programming Through Lambda Calculus**
*by Greg Michaelson, Addison-Wesley, 1988.*
**Reviewer:** Dr. E P Wentworth, Rhodes University

Recently we have seen a number of excellent *second generation* texts on Functional Programming. Michaelson's text assumes some previous programming experience with imperative languages, and presents the functional approach as an alternative paradigm. He begins with a very accessible exposition of the Lambda Calculus, and carefully develops this foundation to encompass the important aspects and paradigms of functional programming. The programming notation is language-independent, although the last chapters are devoted to a brief look at two specific languages, Standard ML and Lisp. The examples and exercises are mainly utility in nature, e.g. "insert a sublist after the first occurrence of another sublist in a list", and can generally be solved in a couple of lines. Answers to the exercises are provided in an appendix.

The approach is slanted towards developing a solid base for understanding functional languages and computing. In this respect the book achieves a good balance between the theoretical underpinnings and their practical application. On the practical side, however, I found the lack of more substantial examples and exercises disappointing. Most programming texts tackle a set of 'standard' problems which are well-understood in the academic community and provide an informal benchmark for comparisons. Since the book is targeted for those already versed in imperative languages and standard algorithms, one might expect the examples to clearly demonstrate the elegance and power of the *problem-oriented* functional approach in these areas. Having laid an excellent foundation I was left with the feeling that the book failed to capitalize and deliver the cherry on the top.

The book is highly recommended as one of the new breed of Computer Science books which gives substantial attention to the fundamentals of the subject without becoming bogged down in over-rigorous formality.

**Artificial Intelligence and the Design of Expert Systems**
*by George F Luger & William A Stubblefield, The Benjamin/Cummings Publishing Co., 1989.*
**Artificial Intelligence: A Knowledge-based Approach**
*by Morris W Firebaugh, PWS-Kent Publishing Co., 1989.*
**Reviewer:** Prof G D Oosthuizen, University of Pretoria

One of the primary goals of an Honours course is to introduce students to a field in such a way that they arrive at enough insight into relevant issues to enable them to conduct further research on their own. To this end a text book which is used ought to reflect the current view of the field. Because of the rapid expansion of the field of Artificial Intelligence (AI), we have now finally outgrown the era dominated by the books by Winston and Charniak and McDermott. In the past five to ten years much new work has been done, and new insights have been gained. Introducing AI, therefore, requires a marked shift from the previous emphasis on a few historical systems embodying a number of famous methods, to a more generic approach - an approach which highlights those fundamental representation and search models that span all the different application areas and strategies of problem solving. Of course, since AI still does not have a well developed theory, references to seminal systems continues to fulfil an important role.

Both of the above books are good text books, characterised by a balanced coverage of Prolog and Lisp. They also reflect and consolidate much of the work of the past few years done in areas such as knowledge representation, machine learning, the work done under the heading of Expert Systems and even the recent work on neural networks. But the most important feature that they share is the accurate and up to date overall picture of the subject provided; the broad framework for the understanding of AI that is created without neglecting work of historical importance. There are still references to these works, but they are placed in perspective in relation to new developments.

The book of Luger & Stubblefield (L&S) is more language oriented than Firebaugh's book. A characteristic of L&S is that AI approaches to representation are related to the Object Oriented approach. Whereas L&S includes chapters on advanced AI programming techniques in Prolog and Lisp, it does not address pattern recognition, computer vision and robotics. (Firebaugh has chapters on each of these themes.) These omissions are understandable, since AI has diversified so much recently that it is difficult to cover all applications in one book.

If I had to select one of the books, it would be L&S. Although L&S gives poor coverage of Machine Learning, the book's overall presentation is very good. In particular, the chapters are well-organised, and the overall approach to AI - starting with the core aspects of *representation* and *search*, followed by chapters on AI languages - is coherent. The authors also make very good use of graphical representations and illustrations to convey ideas.

## Books Received

The following books have been sent to SACJ. Anyone willing to review a book should contact the editor. The book will be sent to him for review, and may be kept provided that a review is received.

- D Bustard, J Elder & J Welsh, [1988], *Concurrent Program Structures*, Prentice-Hall Inc., Englewood Cliffs.
- R Cafolla & A D Kauffman, [1988], *Turbo Prolog Step by Step*, Merrill Publishing Company, Columbus, Ohio.
- S Hekmatpour, [1988], *Introduction to LISP and Symbol Manipulation*, Prentice-Hall Inc., Englewood Cliffs.
- K L Clark & F G McCabe, [1984], *micro-PROLOG: Programming in Logic*, Prentice-Hall Inc., Englewood Cliffs.
- D Crookes, [1988], *Introduction to Programming in Prolog*, Prentice-Hall Inc., Englewood Cliffs.
- M J C Gordon, [1988], *Programming Language Theory and its Implementation*, Prentice-Hall Inc., Englewood Cliffs.
- J G Hughes, [1988], *Database Technology : A software engineering approach*, Prentice-Hall Inc., Englewood Cliffs.
- R Milner, [1989], *Communication and Concurrency*, Prentice-Hall Inc., Englewood Cliffs.
- T J Myers, [1988], *Equations, Models and Programs*, Prentice-Hall, Inc., Englewood Cliffs.
- N C Rowe, [1988], *Artificial Intelligence through Prolog*, Prentice-Hall Inc., Englewood Cliffs.
- D A Protopapas, [1988], *Microcomputer Hardware Design*, Prentice-Hall Inc., Englewood Cliffs.
- H Eisner, [1988], *Computer-aided Systems Engineering*, Prentice-Hall Inc., Englewood Cliffs.
- S H Unger, [1989], *The essence of logic circuits*, Prentice-Hall Inc., Englewood Cliffs.
- R J Young, [1989], *Practical Prolog*, Van Nostrand Reinhold, New York.

## NOTES FOR CONTRIBUTORS

The prime purpose of the journal is to publish original research papers in the fields of Computer Science and Information Systems. However, non-refereed review and exploratory articles of interest to the journal's readers will be considered for publication under sections marked as a 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. Distinguish clearly between such cases as:
  - upper and lower case letters;
  - the letter O and zero;
  - the letter I and the number one; and
  - the letter K and kappa.
- 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:

  [1] E Ashcroft and Z Manna, [1972], The translation of 'GOTO' programs to 'WHILE' programs, *Proceedings of IFIP Congress 71*, North-Holland, Amsterdam, 250-255.

  [2] C Bohm and G Jacopini, [1966], Flow diagrams, Turing machines and languages with only two formation rules, *Comm. ACM*, **9**, 366-371.

  [3] S Ginsburg, [1966], *Mathematical theory of context free languages,* McGraw Hill, New York.

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

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

A page specification is available on request from the editor, for authors wishing to provide camera-ready copies.

**Charges**

A charge per final page, scaled to reflect scanning, typesetting and reproduction costs, will be levied on papers accepted for publication. The costs per final page are as follows:

| | |
|---|---|
| Typed format: | R80-00 |
| ASCII format: | R60-00 |
| Camera-ready format : | R20-00 |

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.

**South African Computer Journal**

**Number 1, January 1990**
**ISSN 1015-7999**

**Suid-Afrikaanse Rekenaar-tydskrif**

**Nommer 1, Januarie 1990**
**ISSN 1015-7999**

# Contents