QUAESTIONES INFORMATICAE

Vol. 1 No. 2

September, 1979

Quaestiones Informaticae

An official publication of the Computer Society of South Africa
'n Amptelike tydskrif van die Rekenaarvereniging van Suid-Afrika

Editors: Dr. D. S. Henderson,
Vice Chancellor, Rhodes University, Grahamstown, 6140, South Africa.
Prof. M. H. Williams,
Department of Computer Science and Applied Maths,
Rhodes University, Grahamstown, 6140, South Africa.

Editorial Advisory Board

Subscriptions

Annual subscriptions are as follows:

	SA	US	UK
Individuals	R2	$3	£1.50
Institutions	R4	$6	£3.00

Quaestiones Informaticae is prepared for publication by SYSTEMS PUBLISHERS (PTY) LTD

for the Computer Society of South Africa.

An Algorithm for the Approximation of Surfaces and for the Packing of Volumes

A. H. J. Christensen

National Research Institute for Mathematical Sciences, CSIR, Pretoria, South Africa

Abstract

An algorithm is presented here that approximates a given surface A by a set of regular or semi-regular faces forming a polyhedron C. If the faces are all square, C assumes the aspect of a 'blocked' surface. It is also possible to say that C is a surface constructed of only a limited number of possible configurations of edges incident with a vertex. It is shown that if square faces are used, C approximates A within $r\sqrt{3}$, where r = length of the edge of the square face. The advantages are discussed of describing A numerically rather than analytically. Technical details of the computer procedure adopted with the algorithm are commented upon. The time complexity is given as kme, where e = number of edges in C and m = number of elements describing A. Applied to closed surfaces, the algorithm can be used to solve the problem of packing solids in a given volume. For an open surface the algorithm produces a closed C that encloses A on all sides.

1. Introduction

The algorithm presented in this paper solves the problem of approximating a given closed surface A by a polyhedric surface C. The closed surface A to be approximated is called here the 'primitive'. The approximation C is a polyhedric surface whose vertices belong to a finite number of classes. Each class includes all vertices in C with the same spatial distribution of incident edges. With some additional simple operations, the algorithm can also be used to solve the problem of packing the volume enclosed by the primitive A with regular or semiregular solids of a limited number of shapes.

Since the solution C will have to approximate a closed surface, it will have to be closed as well. It follows that the simplest approach for an automated procedure would be to choose as surface elements for C the faces of a space-filling solid, which will ensure that all adjacent faces in C will join exactly without gaps.

To further simplify the automated procedure, the solid should be as regular as possible. The two simplest space-filling solids are the cube and a semi-regular set of tetrahedra. The cube as a base for C has the advantage of simplicity; its square faces can be chosen to be parallel to the co-ordinate planes, and all its elements are equal. On the other hand a grid or surface constructed of square faces only is strongly anisotropic and the number of classes of vertices somewhat limited. A surface built up of triangular faces from semiregular tetrahedra exhibits much better homogeneity and is much richer in possible distributions of edges in space. Therefore a surface built up of triangles will approximate the primitive closer and more smoothly, but the corresponding computer procedure, especially the geometry routines, will be far more complex than when square faces are used.

In the algorithm reported here the basic elements are square faces. However most of the routines, i.e. all of them apart from the exceptions mentioned, can be applied to the solution with semi-regular triangular faces. The necessary changes in the main procedure to make it applicable to triangular faces will be mentioned in the text.

As a reader will anticipate and verify from the diagrams at the end of this paper, surfaces built up of only square faces appear to consist of blocks. Hence the use of the terms 'blocked polyhedra' or 'blocked surface'.

The primitive A can be thought of as being embedded in a cubic grid formed by three sets of equidistant planes parallel to the three co-ordinate planes. The distance between parallel planes is r, the resolution of the grid. Therefore the elements of the grid are cubes with edges of length r. On the other hand, the triangular grid is formed by four sets of planes parallel to the four faces of a regular tetrahedron (see [1], p.253).

Surface C will be formed by certain faces of the grid, chosen so that no face of C will intersect A and so that the distance from any face of C to A will be less than $r\sqrt{3}$. An irregularity in the primitive may not be evident in C if its magnitude does not exceed r.

2. The Primitive

The algorithm could accept the primitive A as an analytical surface $A = f(x_i)$; i = 1, 2, 3. In this case the intersection procedure that must be included in the algorithm would have to compute the co-ordinates of the points of intersection between A and edges of the grid, for which purpose the functions

$$A_u = f_u(x_v, x_w);\ u,v,w = 1,2,3;\ u \neq v;\ u \neq w$$

would have to be available to the procedure. This would entail the preparation of a specific subroutine to compute those explicit solutions for every A intended as primitive. Furthermore, many interesting surfaces are not analytical, and to describe them in such a way could in turn introduce greater complexity into the intersection subroutines. Consequently, in order to simplify the algorithm while widening its field of application, the algorithm was prepared for primitives defined numerically. These primitives, again for the sake of simplicity, are expected in the form of triangulations, i.e. each is given by a set of n vertices P with co-ordinates

$x_i(P_j)$; i = 1,2,3; j = 1, . . . ,n

and a set of m triangular faces

$T_i(P_j)$; i =l, . . ., m.

If the primitive is analytical, the algorithm could still be used by first triangularizing A by any of the currently available computer programs, an approach which has the advantage that it obviates both computation of the explicit solutions of A and preparation of the corresponding ad hoc subroutines. However, it must be stressed here that, for certain simple and symmetrical primitives, such as a sphere, in which the three explicit solutions are all analogous, the use of an analytical instead of a numerical description of A in the algorithm greatly simplifies its design and reduces its running time.

3. The Algorithm

In what follows the expression 'cell-intersected' is frequently used. The term indicates that the referred cubic cell has at least one square face which is intersected by one or more of the triangles T. The intersection of a square and a triangle in space is merely a special case of the intersection of two polyhedra, a problem so familiar that it does not merit any additional reference. In this context it is then possible to say that all the cubic cells in the domain of A can be classified as belonging to a class IC of intersected cells or to a class NC of non-intersected cells.

In successive iterations the algorithm constructs C by adding new faces along the open border of the set of faces already established. The new faces are selected according to the conditions mentioned above: they must not intersect any T and the distance from a face to the nearest T must be less than $r\sqrt{3}$.

First a cell C_1 is located which intersects A. C_1 is found by scanning the cells lying against one side of the mini-max box enclosing A and next to the region where A registers an absolute maximum or minimum. If in the neighbourhood of C_1 the primitive is rather flat, four faces of C_1 will be intersected (Fig. 1).

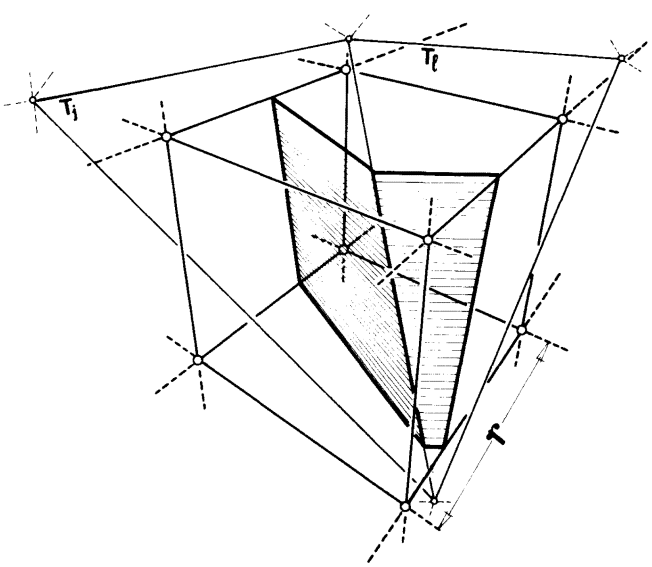


Figure 1. Intersection of triangular faces of T and faces of C_1.

Otherwise, and depending on the orientation and magnitude of the edges of T relative to r in the region, from one to five faces could be intersected. In any case, one of the non-intersected faces, the first to be detected, is chosen as the starting face of C. It is called the 'seed'. The algorithm uses three arrays for edges and one array for faces. Array OE contains the edges in the current cycle open border. In ON are stored the new edges created during the current iteration, which edges will form the open border during the next cycle. Array OP contains the edges belonging to pairs of faces already accepted as part of C. These faces will be stored in array F. All these edge arrays and the face array are empty at first.

The four edges of the seed and pointers to the seed are next stored in array OE. The seed and reciprocal pointers to its edges in OE are stored in array F. At this stage array OE contains four items while array F will contain one, after which the algorithm continues with the following steps for edge OE_i (i = 1):

(1) Edge processing of OE_i. This includes the following.
(a) Retrieval of F_k to which OE_i belongs, by using the reciprocal pointer.
(b) Construction around OE_i of a set of four cells CE. Four faces of CE share OE_i. As will be shown later, each face stored in F belongs to a cell in IC. Therefore one of the cells in CE that shares F_k must also be in IC. It will also be shown later that the other cell sharing F_k is in NC. To indicate this fact, it can be said that F_k is a 'separating' face, because it separates one cell in IC from another cell in NC.

Let cell I (I ∈ IC) and cell IV (IV ∈ NC) be separated by F_k. The other two cells in CE may belong to any of the two classes. They are denoted here by II and III. In the case where a triangular grid is used, a variable number of tetrahedra (between four and six) should be constructed around the edge OE_i. The number depends on the spatial orientation of OE_i.
(c) Search for another separating face F_Q in CE which belongs to OE_i.

There must be at least one such separating face other than F_k. If both II and III are in IC, then F_Q is the other face in IV. This situation is illustrated schematically in Fig. 2(a), where the set of cells CE are projected against a plane perpendicular to OE_i, the four faces having OE_i as common edge being shown as four thick lines, while OE_i is represented by a point. Cells in NC are left blank, while cells in IC are cross-hatched.

If only one of cells II and III is in IC, it must be the one adjoining I (Fig. 2(b)). Otherwise the primitive would either pass through the edge OE_i or it would have two very close branches (closer than r), leaving a gap that the algorithm could not detect (Fig. 2(d)). The first alternative is avoided because the intersection algorithm would have previously classified cell IV in IC as it did with I and then F_k would not have been selected and stored in F. If both II and III are in NC, F_Q will belong to I (Fig. 2(c)). Thus in the cases illustrated in Figs. 2(a), (b) and (c), there is in CE only one separating face F_Q other than F_k. In case (d) there are two such separating faces. The one chosen is that belonging to cell I. The algorithm will find the same relative distribution of separating faces when edge OE_i is encountered again as part of a face created from the right (in Fig. 2(d)). In this case the algorithm will resolve the situation by selecting as F_Q the next separating face to what is now cell I, which is precisely the separating face disregarded when OE_i was first processed. It must be noted that the edge OE_i when processed the second time is in the same spatial location as when first processed, but is an altogether different element in the logical structure describing C.

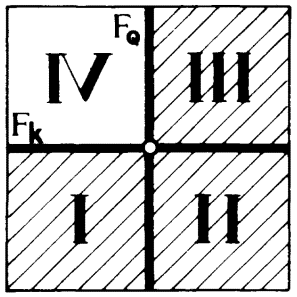


Figure 2(a)

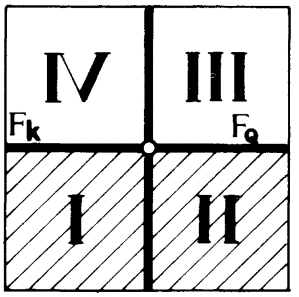


Figure 2(b)

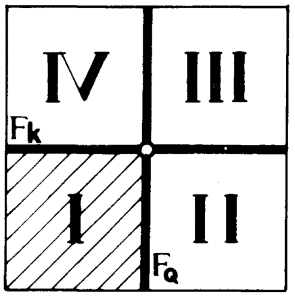


Figure 2(c)

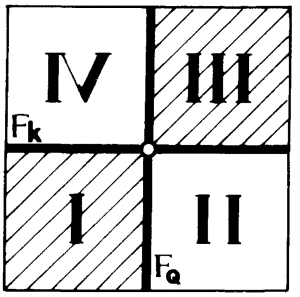


Figure 2(d)

Figure 2(a) to 2(d). Positions of the separating face F_Q.

(d) The separating face F_Q is admitted as part of C and stored in array F, together with pointers to its three edges other than OE_i. These three edges are compared with the edges stored in arrays OE and ON. If they are already in either OE or ON, they are deleted from these arrays and stored in OP, otherwise they are stored in ON.

(e) The process continues for edge OE_{i+1} with step 1. If OE_i was the last element in OE, the process continues with the next step.

(f) Array ON is now considered. If it is not empty, its contents are moved to array OE, and array ON is cleared. Then step 1 and subsequent steps are executed for OE_1. An empty ON array means that there are no more new open edges and that the surface C, fully described by the arrays OP and F, has been completed.

Now it is evident that:

(i) The two cells I and IV in CE must belong to different classes, since all faces in array F are separating faces.

(ii) C will never be further away from A than the distance $r\sqrt{3}$ since all its faces belong to cells in IC.

(iii) C will never intersect any T in the primitive, otherwise some face in F would belong to two cells both in IC and this contradicts corollary (i). Therefore, if C is always close to A (closer than $r\sqrt{3}$) and never intersects it, it can be said with certainty that C approximates A.

4. The Time Complexity and Technicalities

If e is the number of edges in C, it is evident that the time complexity will be proportional to e. And since the processing of one edge involves intersection checks with all T in A, it is also evident that the complexity will be proportional to m as well. The product em is multiplied by a very large factor k, since, as explained, for each edge processed, the algorithm constructs four cells with eighteen faces, all of them but one being checked against the m triangles T. In order to reduce this factor, a procedure was added to the algorithm, which sorts the triangles according to their positions in space and stores them in M large cells (large compared with r). When checking a square face against A, the intersection procedure first positions the square face into the corresponding large cell and then uses only the triangles T stored in that particular large cell. The factor is in this way in the ratio 1/M.

Given a certain amount of computer memory and benefiting from the fact that relatively few elements of C are active simultaneously during the process, it is possible to generate blocked surfaces with many more elements than would have been possible if that fact were ignored. Since only the elements next to the open border are active, the computer procedure that incorporates the algorithm described here uses two sets of double buffers, one for edges and one for faces. The buffers are alternatively filled and emptied; if their lengths are defined correctly, the active elements will always be available in memory. The maximum number of elements active in memory can be estimated as double the number of elements in the longest diameter in C. Since edges are created twice as fast as faces, the edge buffers will have to be twice as long as the buffers for faces.

5. The Generated Surface

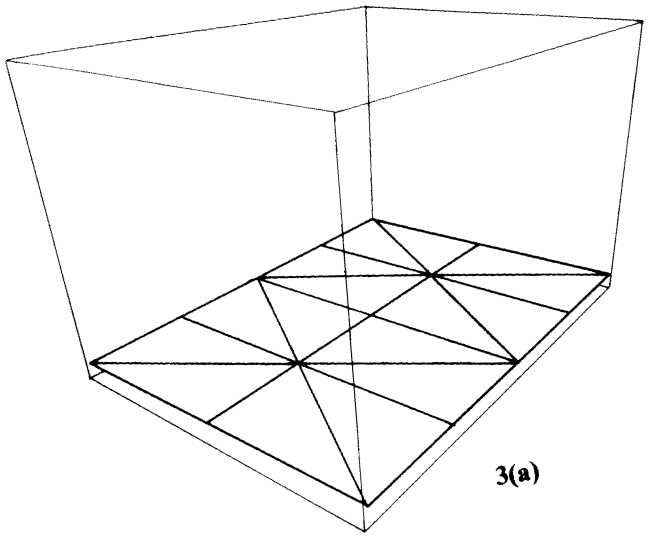
Nothing has been said so far about the position of C with respect to A, i.e. whether C is inside or outside A. Indeed the algorithm has no means of determining this, nor are any data available to it that could be used for such determination, such as an extra point that would not belong to P. Thus two different surfaces C_1 and C_2 can be obtained from the same A, one in its 'inner' space and the other in the 'outer' space. Which surface is used, is decided upon when the first face is selected from the seed cell. If two non-intersected faces are present in the seed, one would originate C_1 and the other C_2. The condition that A be a closed surface is not strictly required for producing C. However, if A is not a closed surface, the approximation would be in a sense double, since C will enclose A on all sides and will not exhibit the holes, open edges or gaps present in A. This behaviour of the algorithm is illustrated in Figs. 3(a) and (b), where A is a plane oblong surface and C has approximated it in the shape of a flat case. If the oblong surface is inclined with respect to the coordinate planes, as in 3(c), the corresponding approximation assumes the form of a double staircase, Fig. 3(d). It is evident then, that applied to open surfaces, the algorithm goes beyond solving the packing problem referred to in the introduction of this paper. Other examples of blocked surfaces are shown in Fig. 4 (sphere) and Fig. 5 (doughnut).

6. Classes and Vertices

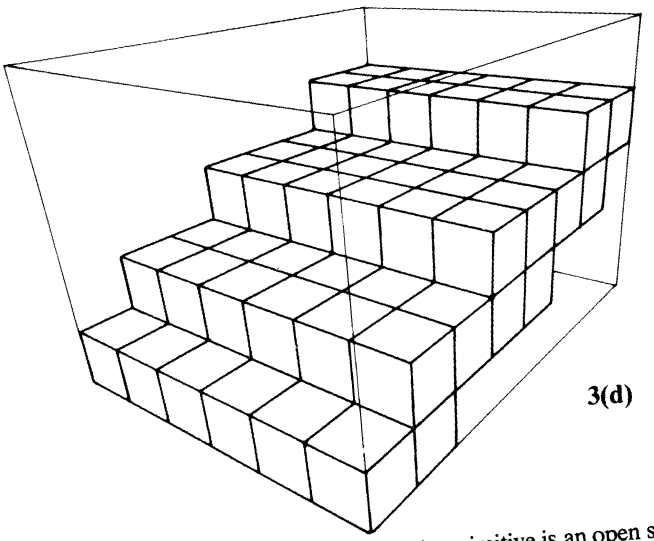
If no distinction is possible between the inside and outside of C, only five spatial configurations of vertices exist in a blocked surface, as follows.

Class A: vertices with three incident edges;
Class B: vertices with four incident edges, all on a plane;
Class C: as for B, but edges at right angles to each other;
Class D: vertices with five edges;
Class E: vertices with six edges.

These five classes of vertices are indicated in Fig. 6. If the number of classes were to be increased, it would be necessary to distinguish between convex and concave configuration of vertices, an extension which would require some definition to distinguish between the inside and outside of C, or the recognition of the configurations of other vertices in the neighbourhood of each one.

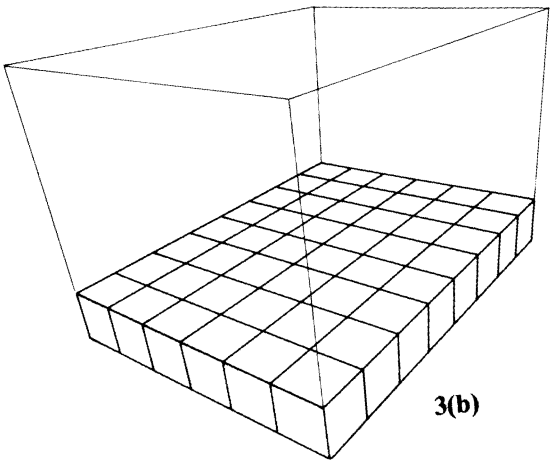


3(a)



3(d)

Figures 3(a) to 3(d). Solutions when the primitive is an open surface.



3(b)

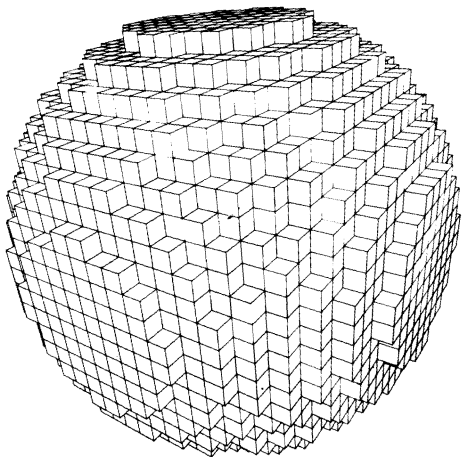
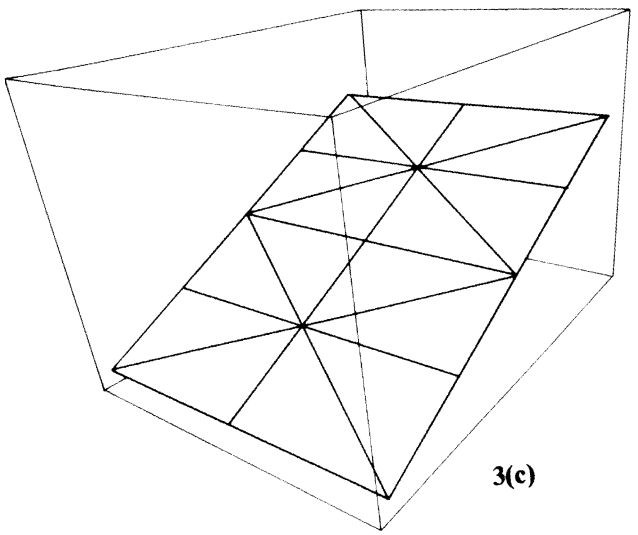


Figure 4. Approximation to a sphere.



3(c)

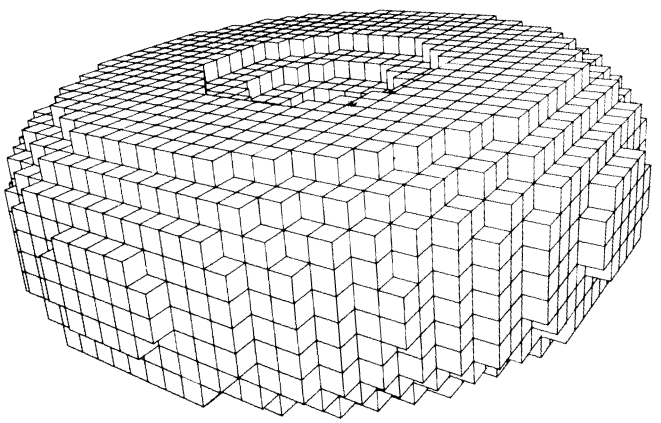


Figure 5. Approximation to a doughnut.

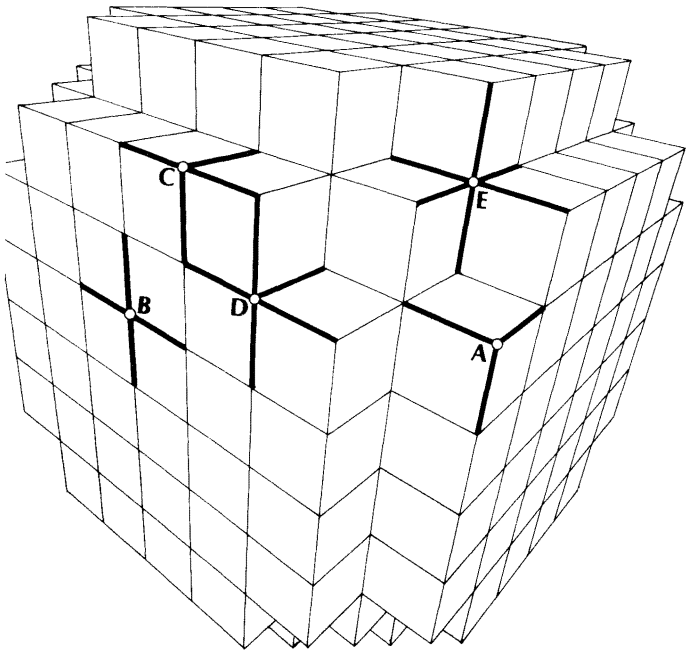


Figure 6. The five classes of vertices.

7. Uses

Having read this far, the reader may possibly be wondering what use could be made of this algorithm or of the blocked surfaces. The writer must confess that he has failed in finding other than the two mentioned in the title of the paper. Nevertheless these surfaces seem to be very intriguing structures. Perhaps they may be used in some new discipline or in extensions of an already established one, such as pattern recognition. The writer is tempted to suggest that the number of vertices in each class could perhaps be used as an indication of the shape of the approximated surface, or that the blocked structures, which can be regarded as 3D digital pictures, could be elements for testing techniques of recognition or representation. Instances of such 3D digital pictures are computer-generated tomograms, see [2], of which blocked surfaces can be considered synthetic or ideally perfect models.

References

[1] GHEORGHIU, A. and DRAGOMIR, V. (1978. Geometry of Structural Forms, Applied Science Publishing Ltd., London.

[2] HERMAN, G.T. and HSUN KAU LU. (1979). Three Dimensional Display of Human Organs from Computed Tomograms, *Computer Graphics and Image Processing,* **9**, 1-21.

Notes for Contributors

The purpose of this Journal will be to publish original papers in any field of computing. Papers submitted may be research articles, review articles, exploratory articles or articles of general interest to readers of the Journal. The preferred languages of the Journal will be the congress languages of IFIP although papers in other languages will not be precluded.

Manuscripts should be in double-spaced typing on one side only of A 4 paper and submitted to Dr. D. S. Henderson or Prof. M. H. Williams at

Rhodes University
Grahamstown 6140
South Africa

Form of manuscript

Manuscripts should be in double-space typing on one side only of sheets of A4 size with wide margins. The original ribbon copy of the typed manuscript should be submitted. Authors should write concisely.

The first page should include the article title (which should be brief), the author's name, and the affiliation and address. Each paper must be accompanied by a summary of less than 200 words which will be printed immediately below the title at the beginning of the paper, together with an appropriate key word list and a list of relevant Computing Review categories.

Tables and figures

Illustrations and tables should not be included in the text, although the author should indicate the desired location of each in the printed text. Tables should be typed on separate sheets and should be numbered consecutively and titled.

Illustrations should also be supplied on separate sheets, and each should be clearly identified on the back in pencil with the Author's name and figure number. Original line drawings (not photoprints) should be submitted and should include all relevant details. Drawings, etc., should be about twice the final size required and lettering must be clear and "open" and sufficiently large to permit the necessary reduction of size in block-making.

Where photographs are submitted, glossy bromide prints are required. If words or numbers are to appear on a photograph, two prints should be sent, the lettering being clearly indicated on one print only. Computer programs or output should be given on clear original printouts and preferably not on lined paper so that they can be reproduced photographically.

Figure legends should be typed on a separate sheet and placed at the end of the manuscript.

Symbols

Mathematical and other symbols may be either handwritten or typewritten. Greek letters and unusual symbols should be identified in the margin. Distinction should be made between capital and lower case letters between the letter O and zero; between the letter l, the number one and prime; between K and kappa.

References

References should be listed at the end of the manuscript in alphabetical order of author's name, and cited in the text by number in square brackets. Journal references should be arranged thus:

1. ASHCROFT, E. and MANNA, Z. (1972). The Translation of 'GOTO' Programs to 'WHILE' Programs, in *Proceedings of IFIP Congress 71,* North-Holland, Amsterdam, 250-255.
2. BÖHM, C. and JACOPINI, G. (1966). Flow Diagrams, Turing Machines and Languages with only Two Formation Rules, *Comm. ACM,* **9,** 366-371.
3. GINSBURG, S. (1966). *Mathematical Theory of Context-free Languages,* McGraw Hill, New York.

Proofs and reprints

Galley proofs will be sent to the author to ensure that the papers have been correctly set up in type and not for the addition of new material or amendment of texts. Excessive alterations may have to be disallowed or the cost charged against the author. Corrected galley proofs, together with the original typescript, must be returned to the editor within three days to minimize the risk of the author's contribution having to be held over to a later issue.

Fifty reprints of each article will be supplied free of charge. Additional copies may be purchased on a reprint order form which will accompany the proofs.

Only original papers will be accepted, and copyright in published papers will be vested in the publisher.

Letters

A section of "Letters to the Editor" (each limited to about 500 words) will provide a forum for discussion of recent problems.

Hierdie notas is ook in Afrikaans verkrygbaar.

Questiones Informaticae

Partial proceedings of the first South African Computer Symposium on Research in Theory, Software, Hardware, organised by The Research Symposium Organising Committee of The Computer Society of South Africa. 4 & 5 September 1979, Pretoria.

Contents/Inhoud