Stars or stripes : a comparative study of finite and transfinite techniques for surface modelling

Eindhoven University of Technology

Department of Mathematics and Computing Science

Stars or Stripes: a comparative study of finite and transfinite techniques for surface modelling

by

C.W.A.M. van Overveld and M. Verhoeven

94/28

Computing Science Note 94/28
Eindhoven, June 1994

COMPUTING SCIENCE NOTES

This is a series of notes of the Computing Science Section of the Department of Mathematics and Computing Science Eindhoven University of Technology.
Since many of these notes are preliminary versions or may be published elsewhere, they have a limited distribution only and are not for review.
Copies of these notes are available from the author.

Copies can be ordered from:
Mrs. M. Philips
Eindhoven University of Technology
Department of Mathematics and Computing Science
P.O. Box 513
5600 MB EINDHOVEN
The Netherlands
ISSN 0926-4515

editors: prof.dr.M.Rem
prof.dr.K.M.van Hee.

Abstract

Most geometric modelling systems employ a surface representation for four-sided patches, such as bi-cubic B-splines, which is based on a matrix of $N \times M$ control points for integer N and M. The boundary is defined by the $2N + 2M - 4$ outermost control points whereas the remaining $MN - 2N - 2M + 4$ control points indicate the shape of the interior part of the surface. This is in sharp contrast with conventional hand-drafting techniques for curved surfaces, where only the boundary is drawn as a set of curve segments joining in the corners, and the shape of the interior part is usually left unspecified. The bi-cubic Coons patches ([Boehm 84]) make up a representation scheme for such surfaces which is much more intuitive in this respect. Yet, bi-cubic Coons patches seem to be not widely used in geometric modelling when compared with B-spline techniques. In this paper, we suggest some causes for this apparent mismatch between intuition and common CAGD-practice *viz.* difficulties of the interactive definition of boundary contours, the specification of cross-boundary derivatives, and a lack of control of the interior shape. Next, we propose remedies for all three types of problems.

Stars or Stripes: a comparative study of finite and transfinite techniques for surface modelling.

C.W.A.M. van Overveld and M.Verhoeven
Department of Mathematics and Computing Science
Eindhoven University of Technology*

May 4, 1994

Keywords

finite and transfinite surface definitions, bi-cubic Coons patches, chain-codes, weaving, CAGD, computer graphics.

1. Introduction

In ([Boehm 84]), a distinction is made between finite and transfinite surface definitions. A *finite* surface definition requires a finite number of parameters, usually in the form of (a rectangular arrangement of) control points, whereas a *transfinite* surface definition is parameterised in terms of continuous functions, generally comprising the contour curves. A typical example of the first class is the B-spline surface; the (bi-cubic) Coons patch represents the second class. In [Boehm 84], the transfinite surface definition receives a

*Address: P.O.Box 513, 5600 MB, Eindhoven, The Netherlands. Email: wsinkvo@info.win.tue.nl, menno@info.win.tue.nl

negligible amount of attention compared to the finite surface definition, and the same seems to hold true for both general texts on computer graphics, current literature on CAGD, and implemented CAGD systems.

Still, the use of continuous contours as defining geometric elements for a surface patch, omitting a graphical indication of the interior shape of the patch, is a much more intuitive match with the ancient tradition of free-hand sketching and drafting techniques than the control-points-based specification of surfaces that has been enforced by the introduction of Bézier and B-spline representations. One may assume, therefore, that the task of designing complex curved surfaces when employing a design tool based on transfinite surface definitions, is less arduous than when employing a conventional control-points-based tool, provided that the same types of operations are supported.

Currently, several difficulties are encountered when founding a CAGD environment on bi-cubic Coons patches:

- an intuitive definition of 3-D contour curves, other than directly specifying arrays of 3-D control points, is not straightforward. In 2-D, the problem is much easier, and most commercial interactive drawing systems support a free-hand curve tool that takes an arbitrary chain of pixels as input and extracts an array of 2-D control points that define a spline curve which approximates the input curve. Algorithms to extract 2-D control points from a 2-D chain curve are well known; an example is [Plass 83]. To specify 3-D contour curves in a similar fashion, however, we have to be able to input and manipulate chains of pixels that contain 3-D information, and to extract continuous curve

representations from these chains.

- B-spline surfaces support a well-defined control over the cross boundary C^1-continuity for adjacent patches by imposing co-planarity constraints on subsets of the control points (the "stars" in the title of this paper). This is easily achieved with bi-cubic Coons patches as well, provided that, aside from the boundary contours, a full set of cross-boundary data is given with the same cardinality as the boundary contours. In other words, instead of four curves, the equivalent amount of information of eight curves should be specified. To put it yet differently, instead of a boundary representation of copper wire (fig. 1a), we need a boundary representation of four connected flat ribbons (fig. 1b; the "stripes" in the title of this paper). This is highly un-practical unless the orientation of the ribbons can be automatically obtained, in such a way that aligned cross boundary derivatives in coincident contour curves guarantee matching cross boundary derivatives in any point along a shared contour curve (fig. 1c).

- The problem of a localised addition of details in the interior of curved surfaces based on control points has been solved by Dave Forsey's introduction of hierarchically defined B-spline patches ([Forsey 88]). Since bi-cubic Coons patches lack any specification of interior detail, a similar device should be available for bi-cubic Coons patches.

This paper is organised as follows:

In section 2, the mathematical framework for representing bi-cubic Coons patches is summarised. A solution to the problem of free-hand editing is

addressed in section 3. It is based on a so called weaving operation on chain-code curves to construct 3-D curves from 2-D connected chains of pixels (section 3.1) and an algorithm for extracting a control point representation from a chain-code curve based on a relaxation technique (section 3.2).

Section 4 discusses the automatic extraction from cross-boundary derivatives for boundary curves. A method for adding interior detail to bi-cubic Coons patches, analogous to Forsey's method, is discussed in section 5, and our conclusions are summarised in section 6.

2. Bi-cubic Coons patches.

An example of a transfinite curved surface representation is the bi-linearly blended Coons patch. It takes as input four curves and produces a curved surface bounded by these curves. Let the four boundary curves be given by $f(0,s), f(r,0), f(1,s)$, and $f(r,1)$. The patch is then given as a function $f(r,s)$. Adjacent patches are in general not G^1, even if the corresponding boundary curves are. G^1 continuity may be provided, however, if a surface is modelled by bi-cubic blending instead of bi-linear blending. In that case the interpolation functions are Hermite polynomials instead of linear functions ([Boehm 84]). This scheme takes as additional input the cross-boundary derivatives $f_r(0,s), f_s(r,0), f_r(1,s)$, and $f_s(r,1)$ and the dual-twist derivatives in the four corners of the modelled surface $f_{rs}(0,0), f_{rs}(0,1), f_{rs}(1,0)$, and $f_{rs}(1,1)$. See [Enderle 84], [Yamaguchi 88], and [Boehm 84] for the mathematical definition of the bi-cubic Coons patch.

3. An intuitive specification technique for boundary curves.

In section 2 Coons patches were seen to take four mutually adjacent boundary curves as input, rather than a collection of control points. In turn, these

boundary curves of course might be specified as an array of 3-D control points, but this clearly wouldn't do justice to the property of transfinity, characteristic of Coons patches. Instead, a method should be employed which supports freehand input to represent any arbitrary 3-D curve. It should be realised, however, that every type of input device (mouse, joystick, track ball, scanner) samples the input trajectories with a finite resolution only. So instead of "arbitrary curves", we have to be able to deal with "arbitrary curves on a 3-D grid" (so called discrete curves). We propose a method to do so, based on weaving (merging) two planar projections of the discrete curve. The idea of weaving is explained further in section 3.1. Next, in order to render the Coons patch, or perform other manipulations, we have to evaluate the boundary curves for arbitrary values of their parameters. Using the discrete curve representation for evaluation would cause discretisation artifacts (aliasing), and therefore the internal curve representation rather should be continuous. In section 3.2 we discuss an algorithm to fit a piecewise 3-D cubic spline curve to a 3-D discrete curve which automatically finds the connection points for the spline segments.

3.1. Weaving discrete curves.

Suppose that first the $X-Y$ projection, P, of the discrete 3-D curve C is drawn. Without loss of generality we assume the point (0,0,0) as the starting point of C, and hence P starts in (0,0). The curve P consists of a series of adjacent grid positions P_i, $P_i \in Z^2$. We adopt a four-connected metric, so $|P_{i+1}.x - P_i.x| = 0$ or $|P_{i+1}.y - P_i.y| = 0$ for $0 \leq i < |P| - 1$, where $|P|$ is the number of points in P. Every vector $P_{i+1} - P_i$ can be represented by a code, p_i, from $\{0, 2, 4, 6\}$, where 0 stands for the vector $(1, 0)$, 2 for $(0, 1)$, 4 for $(-1, 0)$ and 6 for $(0, -1)$. The list of codes p_i is denoted by p;

$0 \leq i < |p|$. This coding scheme is taken from Freeman ([H.Freeman 61a]; see also [H.Freeman 61b], [H.Freeman 69]). Let $P_e = P_{|P|-1}$ represent the endpoint of P. Furthermore, we assume that the curve is monotonous in the positive X-direction, so no 4-codes occur in p. (A non-monotonous curve always can be broken down into monotonous segments).

Next the $X - Z$ projection, Q, is drawn. The codes q_i to represent vectors $Q_{i+1} - Q_i$ are $\{0, y, 4, h\}$, where 0 and 4 have the same meaning as for p, y stands for "yon" (the positive Z-direction), and h stands for "hither" (the negative Z-direction). Again, no 4-codes occur. Q's starting point is also (0,0), but in general, $|P| \neq |Q|$, but since P and Q denote projections of the same 3-D curve, $P_e.x = Q_e.x$; Q_e is Q's end point. Because the chains are hand-drawn, the latter may not hold; this means that Q (or P) has to be preprocessed: it has to be stretched (if $Q_e.x < P_e.x$) or squeezed (if $Q_e.x > P_e.x$). The technique of weaving, to be introduced below, can be used to squeeze or stretch chain-code curves.

In order to reconstruct C from P and Q, we observe that every 0-code in c (i.e. the chain-code of C) projects onto a 0-code in p *and* a 0-code in q. This means that p and q have the same amount of 0-codes. When constructing c from p and q, care should be taken to have the 0-codes synchronised. To compute c, the following steps should be taken:

- For an all-zero fragment in p which corresponds to an all-zero fragment in q of equal length, say l, an all-zero fragment of length l in c is generated.

- For a fragment 0^m (i.e., m zeroes) in p which corresponds to a fragment $0^n q_0 q_1 q_2 \ldots q_{k-1}$ in q, $q_i \neq 0$ and $n < m$, the fragment $0^n q_0 q_1 q_2 \ldots q_{k-1} 0^{m-n}$

is generated, and similarly with the roles of p and q interchanged.

- For a non-zero fragment, say f_p in p, which corresponds to a non-zero fragment, f_q in q, a fragment of length $|f_p| + |f_q|$ has to be output which contains all codes from f_p and all codes from f_q in a uniformly distributed order. We refer to the latter combined fragment as $f_p \mathbf{w} f_q$, and the so-called weaving operator $\mathbf{w}$ is explained below.

An algorithm which meets with these observations is the following (chain-codes are supposed to terminate with a special non-zero dummy code, EOC):

```
chain_code construct_c(p,q)
chain_code p,q;
{
int i=0,j=0;
chain_code c=empty,f_q,f_p;
  while(i<length(p) || j<length(q)) {

/* attempt to create an all-zero fragment */

    while(p[i]=='0' && q[j]=='0'){c=c+'0'; i++; j++;}

/* p[i]!='0' || q[j]!='0'
 * start non-zero fragment */

    f_q=f_p=empty;
    while(p[i]!='0' && p[i]!=EOC) f_p=f_p+p[i++];
```

```
    while(q[j]!='0' && q[j]!=EOC) f_q=f_q+q[j++];

/* weave the non-zero fragments f_p and f_q */

    c=c+weave(f_p,f_q);
  }
  return c;
}
```

To introduce the concept of weaving, we start by assuming that $|f_p| = |f_q|$. Then an obvious choice for the chain-code $f_p \mathbf{w} f_q$ would be

$$f_p \mathbf{w} f_q = f_{p0} f_{q0} f_{p1} f_{q1} f_{p2} f_{q2} f_{p3} f_{q3} \cdots f_{p|f_p|-1} f_{q|f_q|-1},$$

that is, take codes from chains f_p and f_q alternatively. In the general case, however, we have to deal with chain-codes of different lengths. Suppose $|f_p| > |f_q|$. Then we have to take a code from f_q less frequently than a code from f_p. We may regard it as distributing the codes from f_q, as uniformly as possible, over the codes from f_p. This problem is equivalent to interpolating a linear function over a set of equidistant argument values. Now Bresenham's algorithm is a convenient device to do this interpolation. We take a four-connected version of Bresenham's algorithm to produce a chain-code b of the line segment from $(0,0)$ to $(|f_p|, |f_q|)$. For instance, in the case $|f_p| = 2|f_q|$, this chain-code is (spaces have been added for improved readability)

b=020 020 020 020 020 020 020 020 020.....

and if we take a code from f_p every time we encounter a 0 in b and a code from f_q when we encounter a 2 in b, we get the desired distribution. So

in general, the distribution of 0-s and 2-s in a four connected Bresenham line from (0,0) to $(|f_p|, |f_q|)$ to guide the selection of codes from f_p and f_q, gives us the most uniform distribution of codes as possible (see fig. 2 for an example).

In the sequel, the meaning of $p\mathbf{w}q$ is such that the process of selecting codes from p and q is controlled by the 0-s and 2-s from a four-connected Bresenham line from (0,0) to $(|p|, |q|)$.

We observe that the case where $Q_e.x \neq P_e.x$ can also be solved using weaving: let s be the chain $4^{(Q_e.x-P_e.x)}$ in the case $Q_e.x > P_e.x$ or $0^{(P_e.x-Q_e.x)}$ in the case $Q_e.x < P_e.x$, then we first replace q by $q\mathbf{w}s$.

A more complete treatment of (non-uniform) weaving can be found in ([vdW91]). There, also techniques for "cleaning up" chain-codes are discussed, such as annihilation of pairs of opposite codes (e.g. replacing the fragment $x04y$ by xy). In ([Wetering 93]), the problem of editing free-form curves based on weaving is discussed.

3.2. From a discrete curve to a piecewise cubic spline representation

We will first convert the discrete curve C to a poly-line by displacing its vertices, using a relaxation technique, in order to smoothen it; next we fit cubic Bézier segments to well-chosen subsequent segments of the poly-line. Although we illustrate the technique with a 2-D example, we use it for 3-D curves.

Consider the poly-lines in fig. 3. Clearly, fig. 3b is smoother than 3a, but there is still a close resemblance between the two poly-lines: 3a may be regarded as a discretised version of 3b. Indeed, every vertex in 3b is not

removed outside a unit square, centered around the corresponding vertex in 3a. Confining the vertices to unit squares removes quantisation noise (aliasing); using a slightly larger square (say, with size 1.1 or 1.2) gives better results since other high-frequency noise then is removed as well. So we can employ a smoothing algorithm to displace vertices, provided that the vertices are constrained to these squares. A smoothing strategy is readily obtained by stating that for C_{i-1}, C_i, C_{i+1} three subsequent vertices from the poly-line to be smoothed, $|C_i - \frac{C_{i-1}+C_{i+1}}{2}|$ should be minimised. This can be achieved by displacing C_i over a vector $-C_i + \frac{C_{i-1}+C_{i+1}}{2}$, but this affects the centroid of the three points. Rather, we should set for the displacement vectors δ_{i-1}, δ_i, δ_{i+1} for the three C's:

$\delta_i = \frac{2}{3}(-C_i + \frac{C_{i-1}+C_{i+1}}{2})$,

$\delta_{i-1} = \frac{-1}{3}\delta_i$,

$\delta_{i+1} = \frac{-1}{3}\delta_i$.

Since all interior vertices (all vertices except numbers 0, 1, $|C| - 2$, and $|C| - 1$) are engaged in three triples (C_{i-1}, C_i, C_{i+1}) as above, three δ's are computed for all interior vertices. For vertices nr. 0 and $|C| - 1$ we find one δ, and for nrs. 1 and $|C| - 2$ we have two δ's. After computation of all δ's for the original poly-line, the δ's for each vertex are added and the vertex is displaced by the resultant vector. In ([vO93]) it is shown that repeated application of this adjustment converges to a configuration where the vertices are on a discrete cubic curve, provided that all vertices are allowed to move freely, i.e. the vertices are not constrained to the unit squares as explained above. In case they are, some vertices will be affected by this constraint (to be called k-vertices), whereas others aren't. Now we observe that the non-k-

vertices, between two subsequent k-vertices, are again allowed to move freely during the relaxation process, and hence, after convergence, they will lie on a discrete cubic curve segment, whereas adjacent discrete cubic curve segments join in adjacent k-vertices. So the k-vertices will denote the locations of the extreme control points of the resulting cubic Bézier segments. For each Bézier segment, the remaining two control points are easily fitted to the intermediate non-k-vertices, thereby taking C^1 continuity into account by demanding subsequent sequences of three control points, as in fig. 4, to be co-linear.

This completes the conversion from a discrete curve representation for the hand-drawn curve to a piecewise cubic curve; when implementing the relaxation technique, a multi-grid method should be employed to increase the convergence speed (i.e., first convergence should be achieved on a resolution, say, a factor of 32 less than the final resolution; next a factor of 16 less, next 8, and so on).

4. A method for computing cross boundary (cb) and dual twist (dt) derivatives.

We state the following requirements, to be met by our method:

1. If boundary curves of adjacent patches are G^1, the patches must border in a G^1 manner.

2. If a rotation sweep surface exists, compliant with the boundary curves, in the sense that two opposite curves are circular arcs whereas the other two curves are congruent, then the computed cb and dt-derivatives are compliant with this surface. Examples of such a configuration are depicted in fig.-s 5a, 5b, and 5c.

3. If the curves all lie in a common plane, then the derivatives must be parallel to this plane.

4. If the curves are subjected to affine transformations the derivatives must be transformed accordingly.

We observe that the directions of the cb-derivatives are the only factors affecting G^1 borders of adjacent patches. Therefore, requirement 1 can be seen to be equivalent to:

- The direction of the cb-derivatives, e.g. $f_s(r,0)$, must be uniquely determined by the curve itself ($f(r,0)$) and the tangents of the curves incident to its endpoints ($f_s(0,0)$ and $f_s(1,0)$).

- The direction of the cb-derivatives in any point of a boundary curve must be independent of the length of the cb-derivatives in the corners of the boundary. So we make sure that only the normalised cb-derivatives in the corners of the curves are used when determining the direction of the cb derivatives.

The method can now be split into three parts: computation of the direction of the cb-derivatives, computation of the lengths of the cb-derivatives, and computation of the dt-derivatives.

4.1. Computation of the direction of cb-derivatives

Let $f(t)$ be a boundary curve and e_0 and e_1 are the normalised cb-derivatives in $f(0)$ and $f(1)$, respectively. In case the boundary curves are compliant with a rotation sweep surface of which $f(t)$ is a circular sweep-arc segment,

the lines l_0 (through e_0) and l_1 (through e_1) intersect, and the intersection point g is the apex of the circular cone[1], described through $f(t)$. So in this case, $f_{cb}(t)$ should be a unit vector originating in $f(t)$, directed along the line $f(t) - g$. In other words, the cb-derivatives are on a conic mantle with g as apex, which passes through $f(t)$. Now for a general configuration, we propose to define a generalisation of the point g, such that again $f_{cb}(t)$ can be defined as the unit vector, originating in $f(t)$, directed along $f(t) - g$. A naive generalisation for g is the midpoint of the shortest straight line segment which connects l_0 and l_1, but then $f_{cb}(0) \neq e_0$ and $f_{cb}(1) \neq e_1$. Instead of a point g, however, we can the shortest line segment, $g(t)$, with $g(0)$ on l_0 and $g(1)$ on l_1, thus $g(t)$ is perpendicular on both l_0 and l_1. This gives $f_{cb}(t)$ the direction $f(t)-g(t)$, so the cb-derivatives are on a generalised conic mantle with $g(t), 0 \leq t \leq 1$ as a ridge (apex-line) (see fig. 6). Note that, if l_0 and l_1 intersect, $g(t), 0 \leq t \leq 1$ again reduces to one point.

Some precautions must be taken:

- The cb-derivative has direction either $f(t)-g(t)$ or $g(t)-f(t)$; a choice between the two is not a priori clear;
- $g(0)$ and $f(0)$ may coincide, and similar for $g(1)$ and $f(1)$;
- if e_0 and e_1 are parallel, g cannot be computed directly.

A complete treatment of these issues is outside the scope of this paper; see [Verhoeven 92] for a more detailed account of the computation of $g(t)$ and the direction of the cb-derivatives.

[1] If e_0 and e_1 are parallel, this apex lies in infinity and a cylinder results.

4.2. Computation of the length of cb derivatives

Whereas the directions of the cb-derivatives have to depend solely on $f(t)$ and e_0 and e_1 from sect. 4.1., their lengths may depend on other parts of the boundary input as well, e.g. the boundary curve opposite from $f(t)$, say $f_{opp}(t)$. In particular, in the case of rotation sweep surfaces, $\frac{|f_{cb}(t)|}{|f(t)-f_{opp}(t)|}$ should be constant (see fig. 7). For the cb-derivatives from the input curves, however, $f_{cb}(0)$ and $f_{cb}(1)$, it is not necessarily true that $\frac{|f_{cb}(0)|}{|f(0)-f_{opp}(0)|} = \frac{|f_{cb}(1)|}{|f(1)-f_{opp}(1)|}$. So a somewhat weaker formulation to compute $|f_{cb}(t)|$ is, again using linear interpolation:

$$\frac{|f_{cb}(t)|}{|f(t) - f_{opp}(t)|} = \frac{|f_{cb}(0)|}{|f(0) - f_{opp}(0)|}(1-t) + \frac{|f_{cb}(1)|}{|f(1) - f_{opp}(1)|}t$$

Intuitively, this formula states that the length of a $f_{cb}(t)$ (cf. the speed) increases if the distance between opposite sides of the face increases.

4.3. Computation of the dt-derivatives

Two different methods can be used to determine dt-derivatives: either, differentiate f_r to s, or differentiate f_s to r. In general, these two possibilities do not yield the same result and there is no way to determine which of the two approaches is best for which cases. The simplest symmetric solution is to average both expressions. A complete derivation is given in [Verhoeven 92].

5. Adding detail to bi-cubic Coons patches.

In order to solve the problem of adding detail to B-spline patches, Forsey ([Forsey 88]) suggested to allow the user to control the location of control points that are generated during recursive subdivision of the B-spline patch. If we assume bi-cubic Coons patches to arise from a subdivision process, we can apply a similar technique to add detail here.

The most natural subdivision scheme for B-spline patches is to generate 4 sub-patches for each iteration. So in case of a bi-cubic B-spline patch we start with 16 control points, and after one subdivision we have 4 sets of 16 control points. The user may decide to edit zero or more of these sets, and similar for their recursive descendants.

The similar procedure for bi-cubic Coons patches is as follows. Let $f(r,0)$, $f(r,1)$, $f(0,s)$, and $f(1,s)$ and their derivatives define the patch $f(r,s)$, for $0 \leq r \leq 1$, $0 \leq s \leq 1$. This patch can be thought of to be composed from two sub-patches, say defined by

$f(r,0)$, $f(r,1/2)$, $f(0,s)$, $f(1,s)$, for $0 \leq r \leq 1$, $0 \leq s \leq 1/2$;

and

$f(r,1/2)$, $f(r,1)$, $f(0,s)$, $f(1,s)$, for $0 \leq r \leq 1$, $1/2 \leq s \leq 1$,

provided all $s-$derivatives are scaled with a factor 2. In this sense, we may say that we split the patch in two sub-patches in the $s-$direction. Joining these two sub-patches gives the original patch $f(r,s)$, so in this case we have added no detail. But we may wish to modify the curve $f(r,1/2)$, in a similar way as we may modify some of the control points that occur during the subdivision process in Forsey's method. As an additional degree of freedom, we may even decide to split the original patch for a value of s different from 1/2; also, we may split the patch in the r-direction instead. Fig. 8 shows some stages of the process of designing a mask using this technique. We summarise the main characteristics of this method:

- In Forsey's method, only those control points are edited that contribute to shape details that differ from the original shape of the patch. Here,

only those curves are drawn that add shape details that differ from the original shape of the patch.

- In Forsey's method, each subdivision takes place in four sub-patches according to the standard B-spline subdivision scheme, so a regular quadtree arises in parameter space. In our method, a binary tree results that is not necessarily regular (see fig. 9).

- In order to render the patches, they should be converted into a polygon mesh. To avoid cracks, the (non-)regular binary tree of bi-cubic Coons patches is completed into a regular mesh (see fig. 10). Given the tree structure, this conversion can take place automatically.

6. Summary; discussion

When comparing finite and transfinite methods for surface modelling, it turns out that transfinite methods correspond more closely to traditional drafting practice; still, they are used less frequently. Among other things, this could be caused by difficulties in defining 3-D curves in an intuitive manner, difficulties in specifying cross boundary derivatives, and/or difficulties in specifying interior shape detail. Provided these difficulties are adequately solved, bi-cubic Coons patches may form a suitable alternative for control points-based patches. To evaluate their merits, we ran two tests. First, we attempted to model a humanoid face using little user input according to the strategy of fig. 8. Fig. 11 shows 3-D poly-line approximations to the contours for a 6-step design session. The associated solid rendered masks are depicted in colour plate I. It turns out that the definition of just 11 curves, which is few enough to keep the wire frame representation from getting cluttered, suffices to produce a recognisable final rendered result. As

a second test, we compare the canonical Bézier patch representation of the Utah teapot with a representation of the same model in terms of bi-cubic Coons patches. In fig. 12, the lower left object is a poly-line representation of the control polygons of the Bézier patches of the original teapot. To get a fair comparison we have omitted the internal 4 control points for each patch in the model, and only the 12 boundary control points are used. We observe that the control polygons form a rather crude approximation of the smooth shapes of the teapot; especially the spout and the handle are difficult to recognise. The lower right image in fig. 12 is the set of contour curves, used to define bi-cubic Coons patches modelling the same object. These curves were obtained by sampling the cubic Bézier curves that form the borders of the bi-cubic Bézier patches in the original model. In the wire frame, the shape of the teapot is much easier recognised whereas in the solid rendering in colour plate II hardly any visible differences between the two tea pots show up. As an additional bonus, the specification in terms of contours rather than control points allows us to apply subtle local deformations: observe the bulging handle, the perturbed spout and the shape details in the lid and the upper rim in the topmost version of the teapot in fig. 12 and colour plate II.

References

[Boehm 84] W. Boehm, G. Farin, and J. Kahman. A survey of Curve and Surface methods in CAGD. *Computer Aided Geometric Design*, 1:1–60, 1984.

[Enderle 84] G. Enderle, D. Duce, and P.J.W. ten Hagen (Eds). *Eurographics Tutorials 1983*. The European Association for Computer Graphics, 1984.

[Forsey 88] David R. Forsey and Richard H. Bartels. Hierarchical B-spline refinement. *Computer Graphics (Proc. SIGGRAPH 88)*, 22(4):205–212, August 1988.

[H.Freeman 61a] H.Freeman. On the encoding of arbitrary geometric configurations. *IRE Transactions on Electronic Computers*, pages 260–268, June 1961.

[H.Freeman 61b] H.Freeman. Techniques for the digital computer analysis of chain-encoded arbitrary plane curves. *Proceedings of the National Electronics Conference, Chicago*, 17:421–432, 1961.

[H.Freeman 69] H.Freeman. A scheme for the efficient encoding of graphical data for communication and information processing. *Advance in Electronics, proceedings of the 16th electronic congres, Rome*, pages 340–348, March 1969.

[Plass 83] Michael Plass and Maureen Stone. Curve Fitting with Piecewise Parametric Curves. *ACM Computer Graphics*, 17(3):229–239, July 1983.

[vdW91] H.M.M. van de Wetering. *Chain Coding in Computer Graphics*. PhD thesis, Eindhoven University of Technology, Dept. of Mathematics and Computing Science, November 1991.

[Verhoeven 92] Menno Verhoeven. *Modelling with bi-cubic Coons Patches (in dutch).* internal report Eindhoven University of Technology, Department of Mathematics and Computing Science, EUT,Eindhoven, the Netherlands, 1992.

[vO93] C.W.A.M. van Overveld. The application of relaxation and optimisation methods in computer aided geometric design. *Proceedings of ATARV-93:Advanced Techniques in Animation, Rendering and Visualisation; B.Özgüç and V. Akman, eds., Ankara, Turkey*, pages 161–180, July 1993.

[Wetering 93] H.M.M. van de Wetering and C.W.A.M. van Overveld. Chain Codes and their Application in Curve Design. *Intern report; Eindhoven University of Technology, Dept. of Mathematics and Computing Science (to be published)*, October 1993.

[Yamaguchi 88] F. Yamaguchi. *Curves and Surfaces in Computer Aided Design.* Springer Verlag, Berlin Heidelberg, 1988.

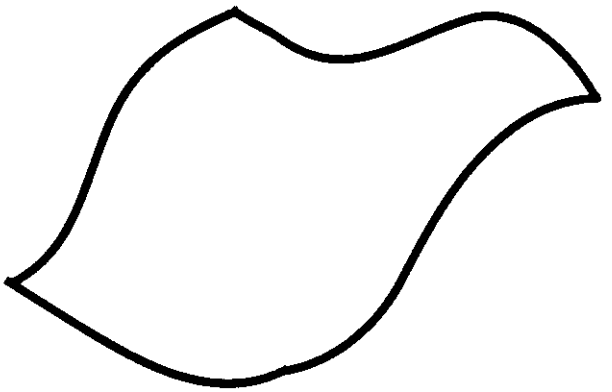

Fig. 1a: boundary specification with four curves.

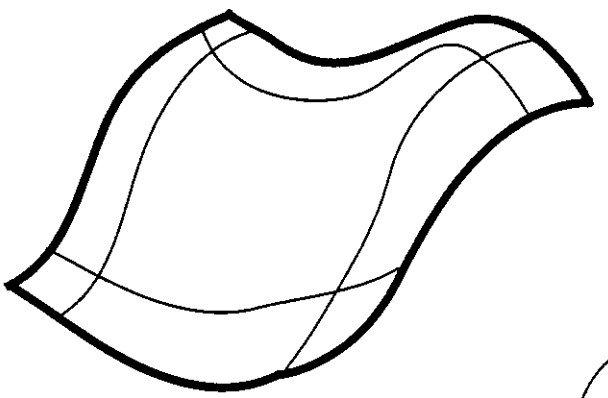

Fig. 1b: boundary specification with four curves plus cross boundary derivatives.

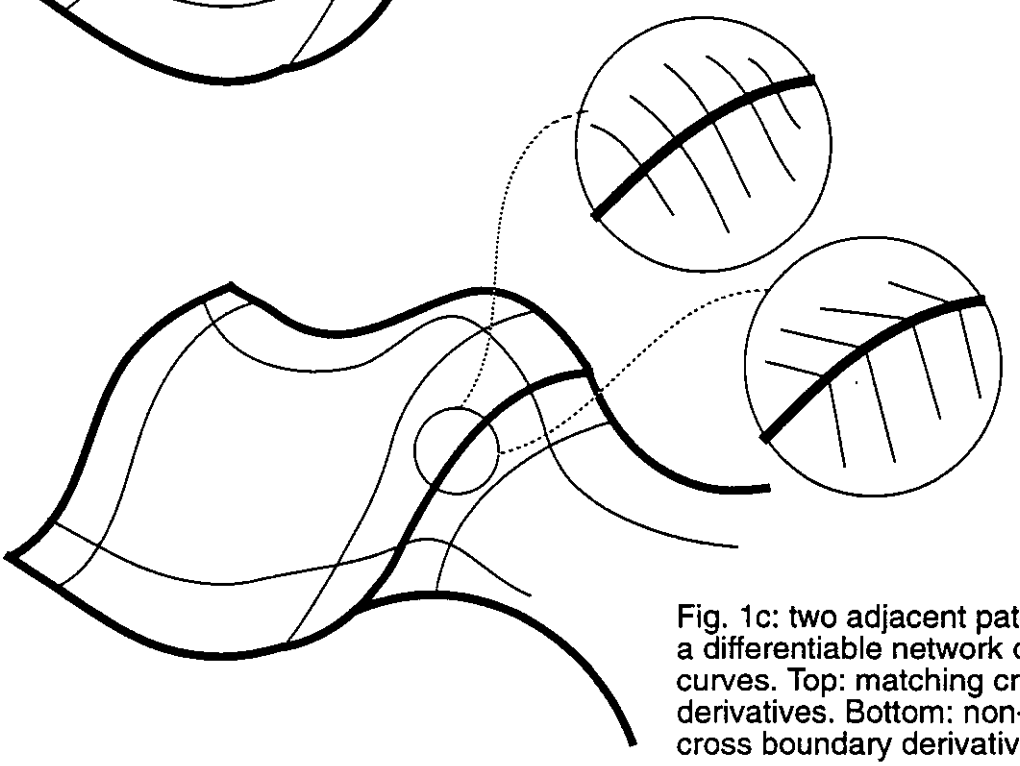

Fig. 1c: two adjacent patches with a differentiable network of boundary curves. Top: matching cross boundary derivatives. Bottom: non-matching cross boundary derivatives.

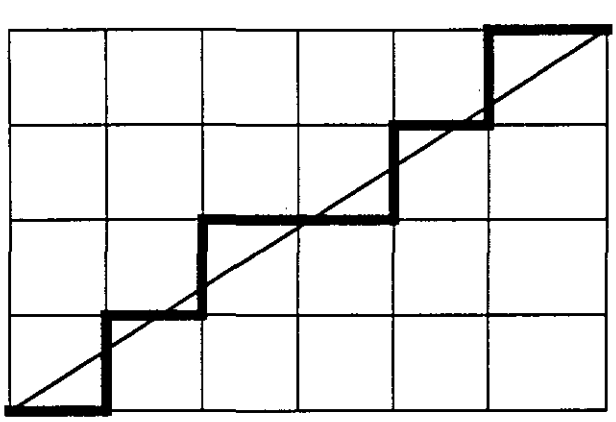

Fig. 2
Weaving chains c (codes CC...) and d (codes DD...) takes place using the distribution of codes 0 and 2 in the Bresenham chain, b, of the line from (0,0) to (|c|,|d|). A 0-code forces the next element from c, a 2-code the next element from d.

b=0202002020

c=CCCCCC

d=DDDD

cWd=CDCDCCDCDC

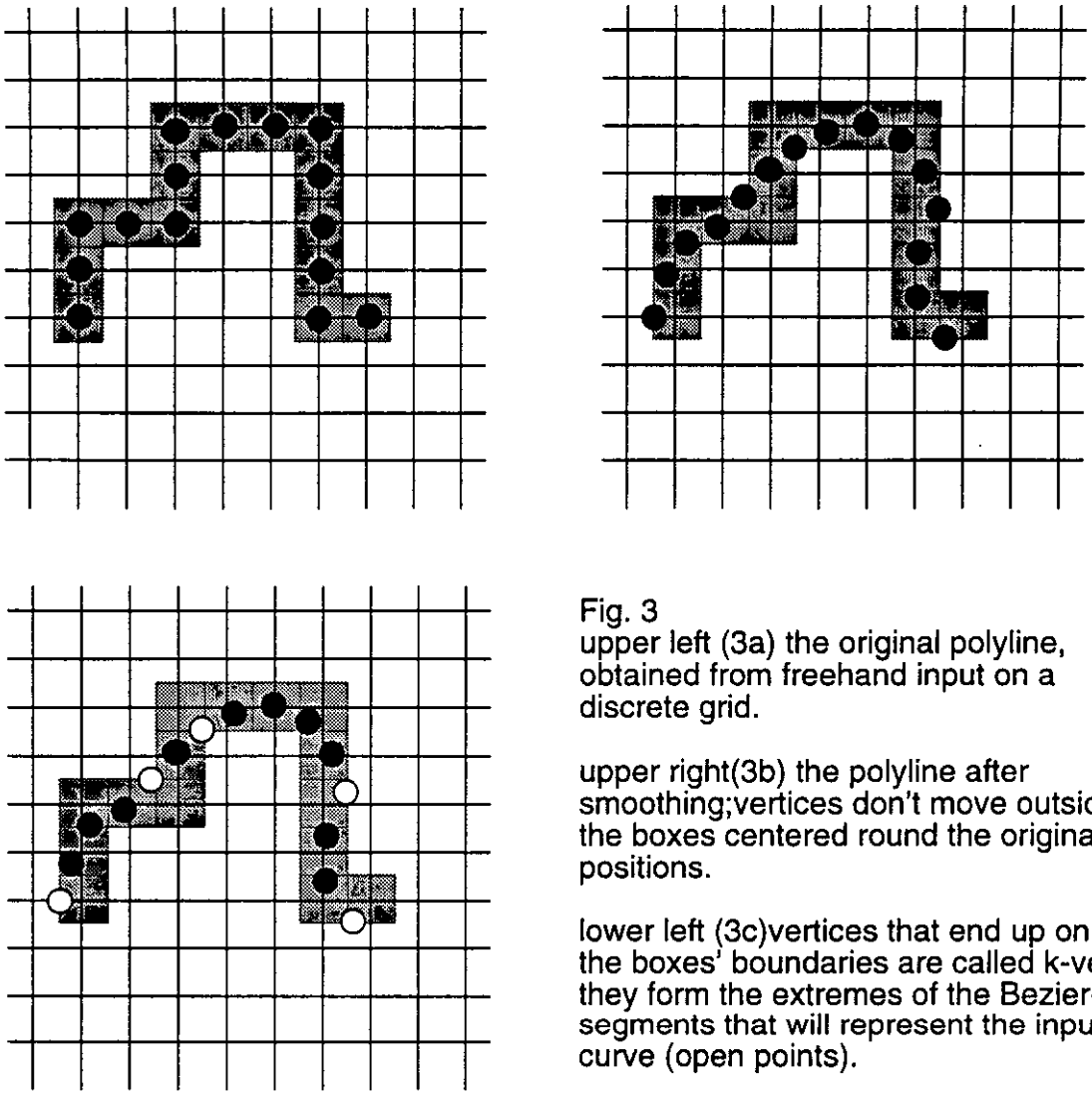

Fig. 3
upper left (3a) the original polyline, obtained from freehand input on a discrete grid.

upper right(3b) the polyline after smoothing;vertices don't move outside the boxes centered round the original positions.

lower left (3c)vertices that end up on the boxes' boundaries are called k-vertices; they form the extremes of the Bezier-segments that will represent the input curve (open points).

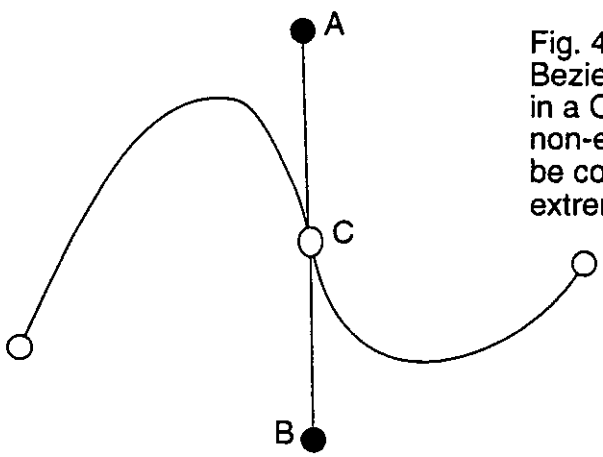

Fig. 4
Bezier spline segments are connected in a C1 -fashion by adjusting the non-extreme control points (A,B) to be co-linear with the corresponding extreme (C).

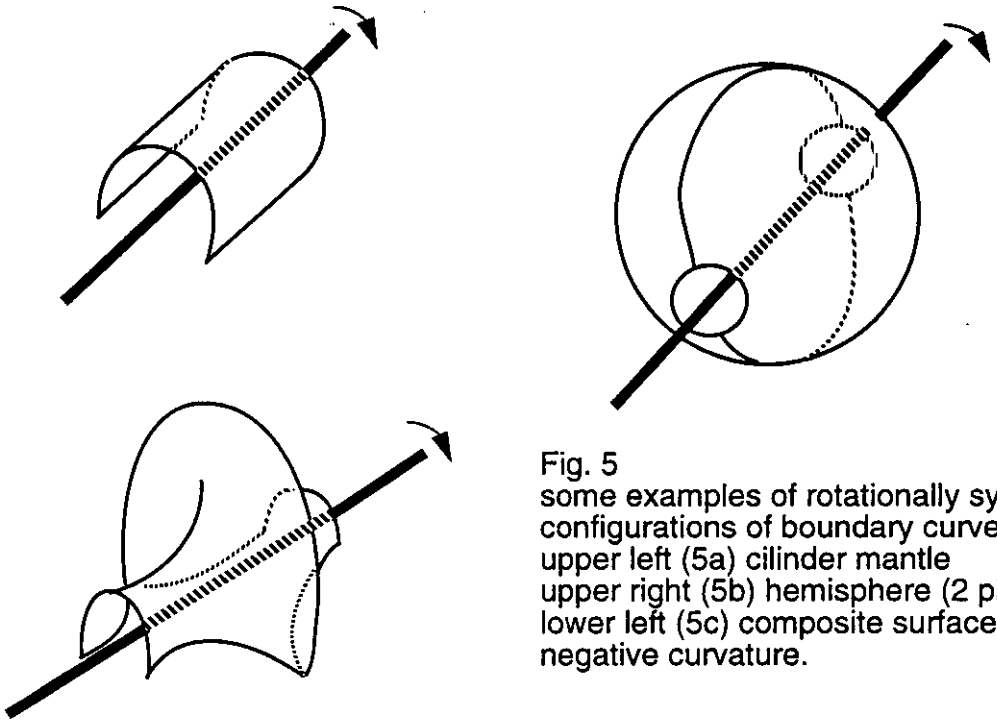

Fig. 5
some examples of rotationally symmetric configurations of boundary curves.
upper left (5a) cilinder mantle
upper right (5b) hemisphere (2 pieces)
lower left (5c) composite surface with negative curvature.

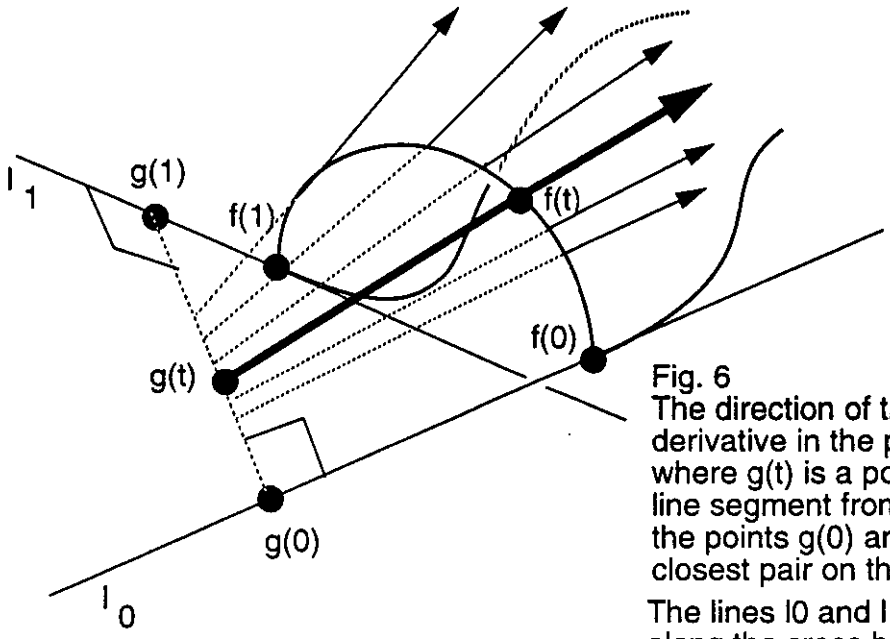

Fig. 6
The direction of the cross boundary derivative in the point f(t) is f(t)-g(t) where g(t) is a point on the straight line segment from g(0) to g(1); the points g(0) and g(1) are the closest pair on the lines l0 and l1.

The lines l0 and l1 are directed along the cross boundary derivatives in f(0) and f(1).

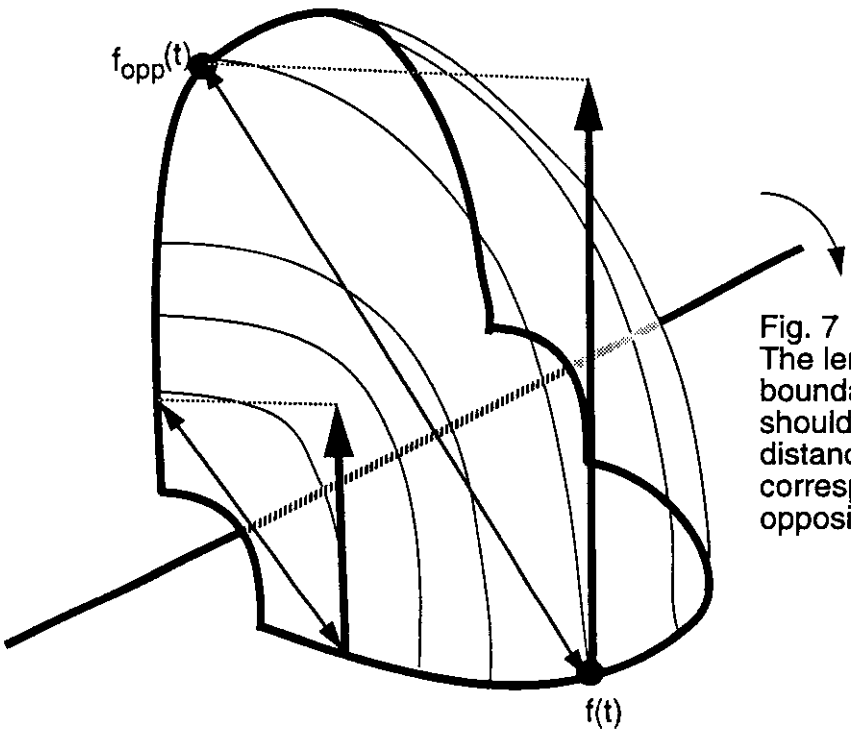

Fig. 7
The length of the cross boundary derivative in point f(t) should be proportional to the distance between f(t) and the corresponding point on the opposite curve, f_{opp} (t).

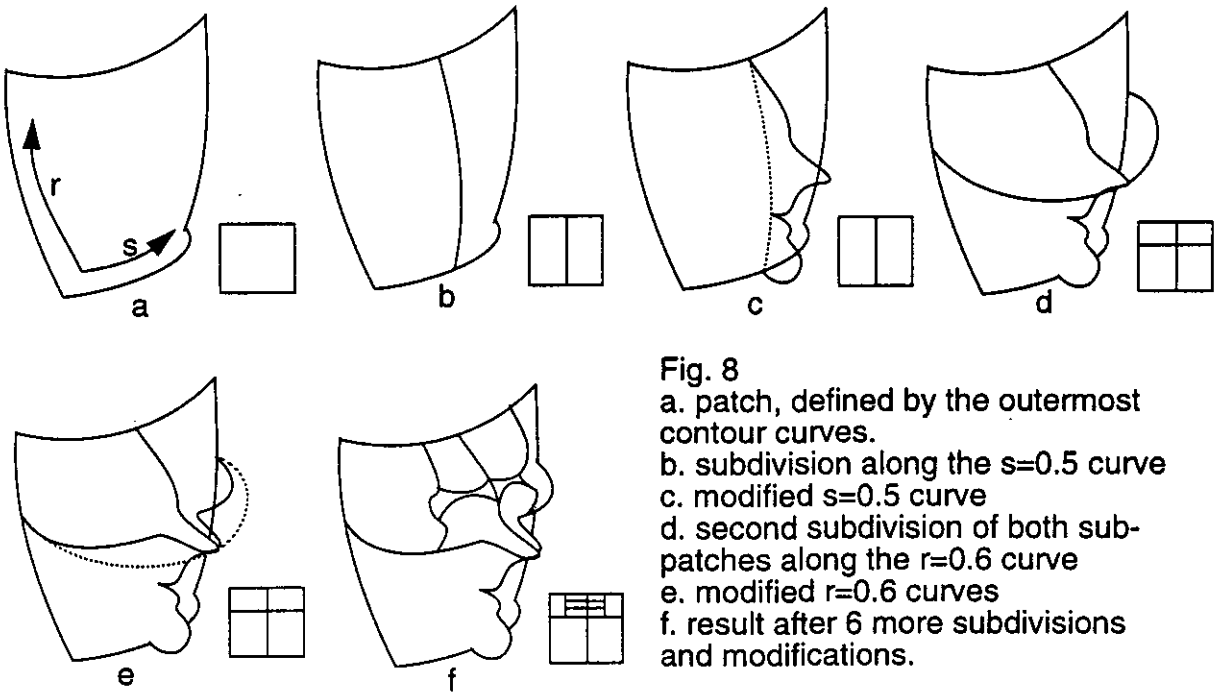

Fig. 8
a. patch, defined by the outermost contour curves.
b. subdivision along the s=0.5 curve
c. modified s=0.5 curve
d. second subdivision of both sub-patches along the r=0.6 curve
e. modified r=0.6 curves
f. result after 6 more subdivisions and modifications.

The subdivisions in parameter space are shown in the lower right diagrams.

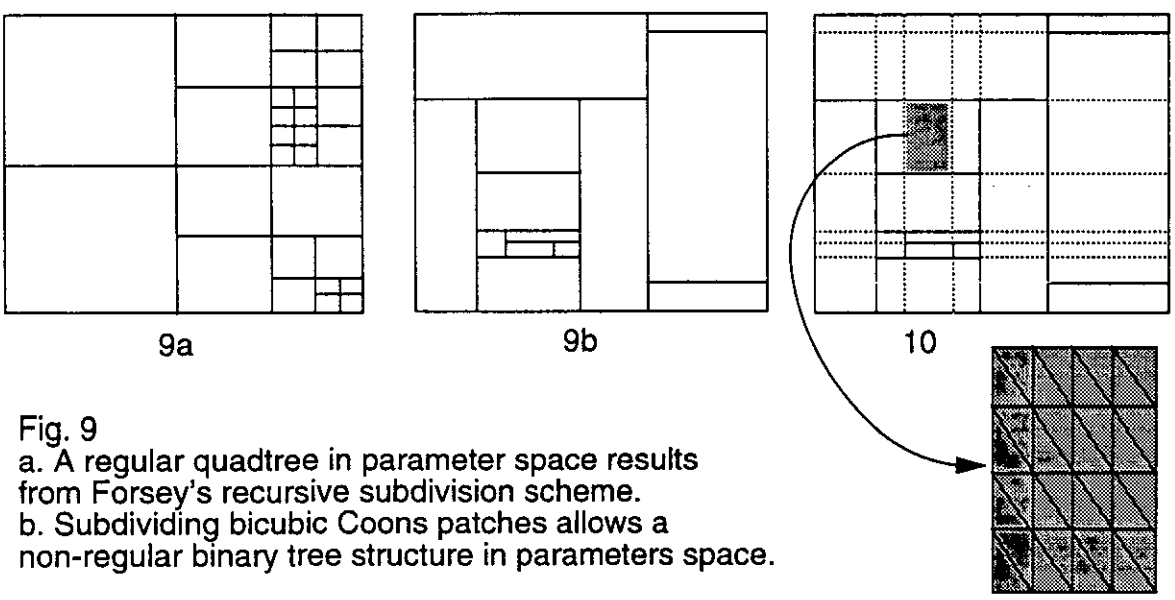

Fig. 9
a. A regular quadtree in parameter space results from Forsey's recursive subdivision scheme.
b. Subdividing bicubic Coons patches allows a non-regular binary tree structure in parameters space.

Fig. 10
To avoid the crack problem, the non-regular binary tree structure can be completed to form a (non-regular) rectangular grid topology. Inset: each Coons patch is converted into a polygon mesh before rendering.

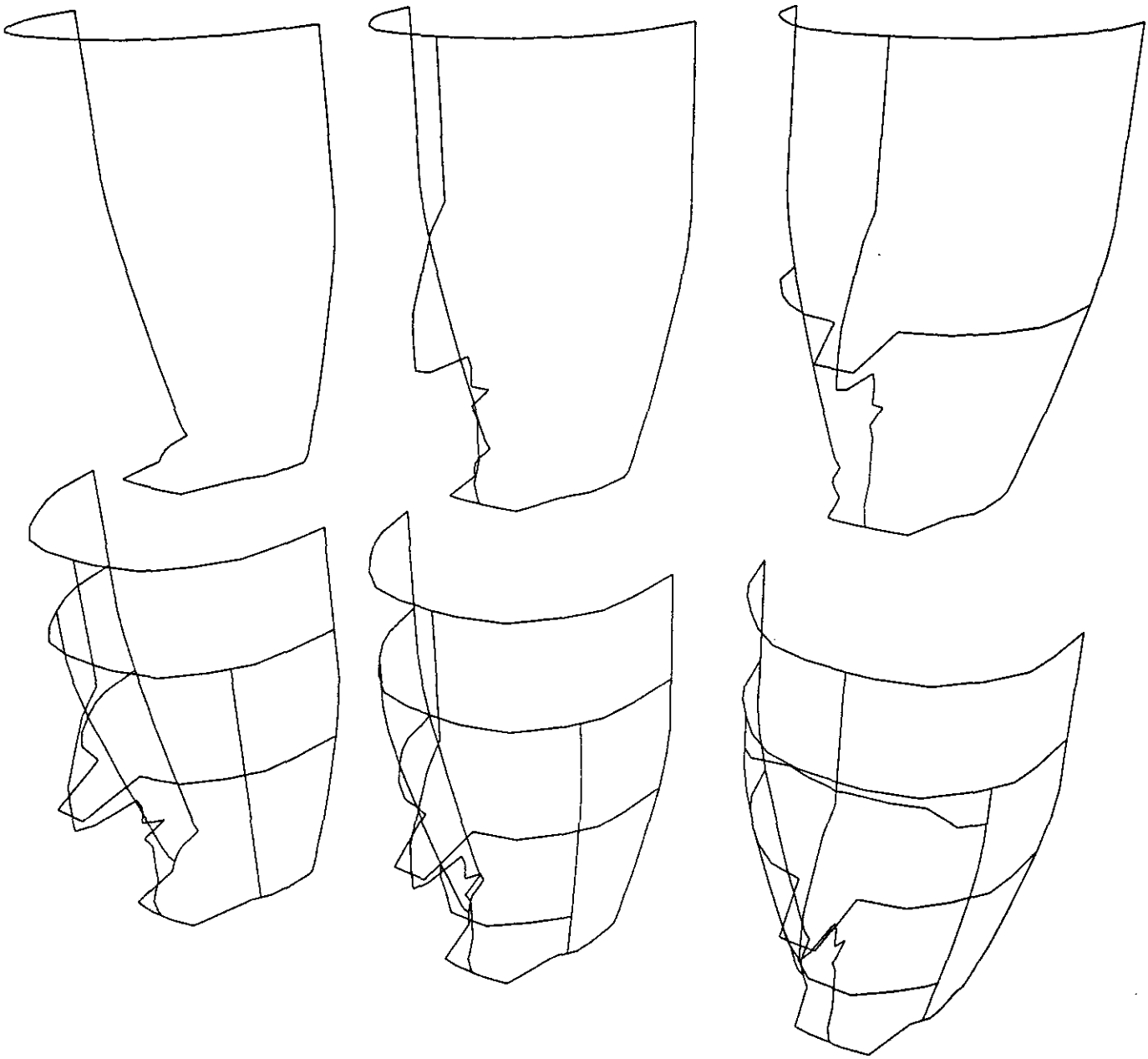

Fig. 11

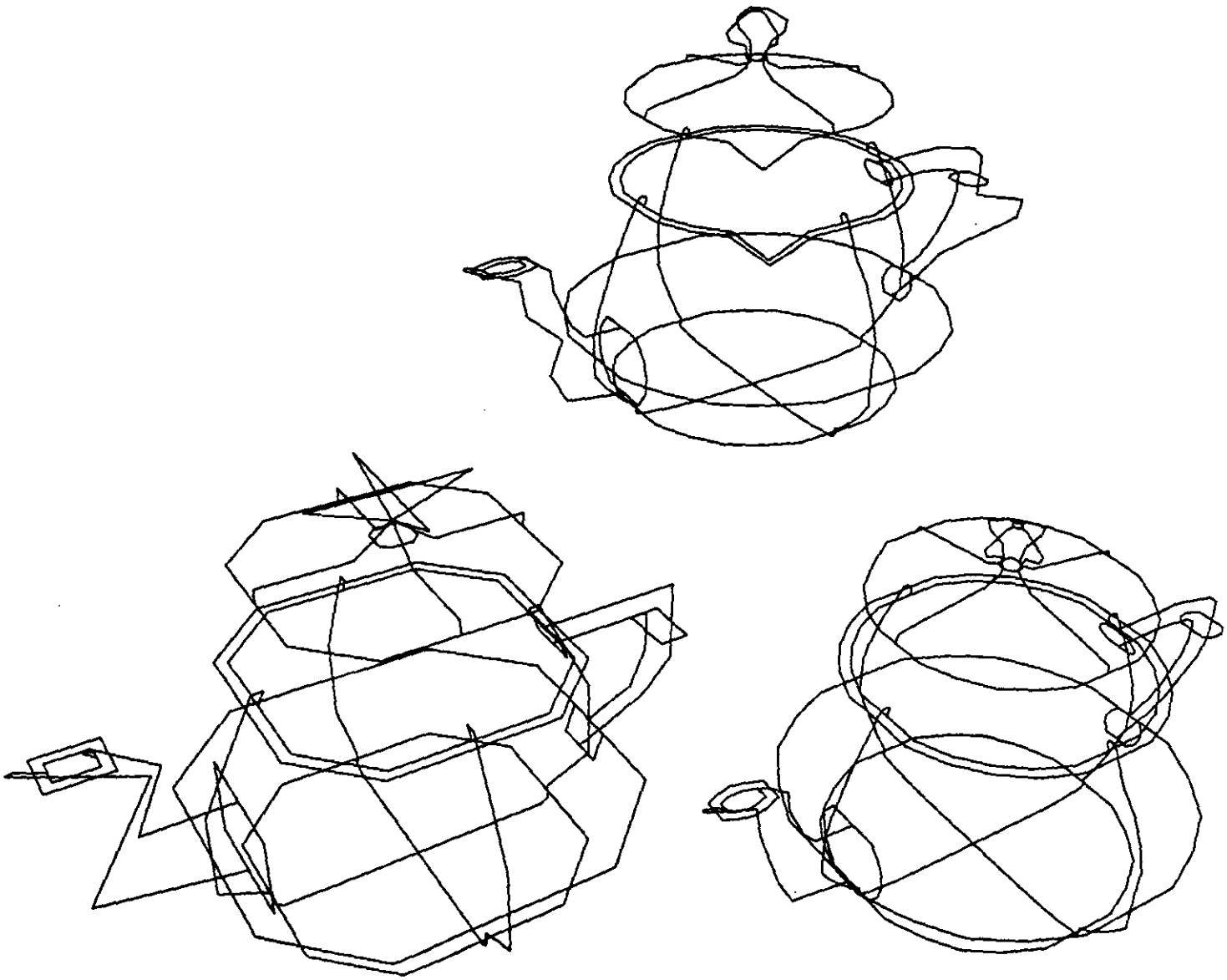

Fig. 12

Computing Science Notes

Department of Mathematics and Computing Science
Eindhoven University of Technology

In this series appeared:

91/01	D. Alstein	Dynamic Reconfiguration in Distributed Hard Real-Time Systems, p. 14.
91/02	R.P. Nederpelt H.C.M. de Swart	Implication. A survey of the different logical analyses "if...,then...", p. 26.
91/03	J.P. Katoen L.A.M. Schoenmakers	Parallel Programs for the Recognition of *P*-invariant Segments, p. 16.
91/04	E. v.d. Sluis A.F. v.d. Stappen	Performance Analysis of VLSI Programs, p. 31.
91/05	D. de Reus	An Implementation Model for GOOD, p. 18.
91/06	K.M. van Hee	SPECIFICATIEMETHODEN, een overzicht, p. 20.
91/07	E.Poll	CPO-models for second order lambda calculus with recursive types and subtyping, p. 49.
91/08	H. Schepers	Terminology and Paradigms for Fault Tolerance, p. 25.
91/09	W.M.P.v.d.Aalst	Interval Timed Petri Nets and their analysis, p.53.
91/10	R.C.Backhouse P.J. de Bruin P. Hoogendijk G. Malcolm E. Voermans J. v.d. Woude	POLYNOMIAL RELATORS, p. 52.
91/11	R.C. Backhouse P.J. de Bruin G.Malcolm E.Voermans J. van der Woude	Relational Catamorphism, p. 31.
91/12	E. van der Sluis	A parallel local search algorithm for the travelling salesman problem, p. 12.
91/13	F. Rietman	A note on Extensionality, p. 21.
91/14	P. Lemmens	The PDB Hypermedia Package. Why and how it was built, p. 63.
91/15	A.T.M. Aerts K.M. van Hee	Eldorado: Architecture of a Functional Database Management System, p. 19.
91/16	A.J.J.M. Marcelis	An example of proving attribute grammars correct: the representation of arithmetical expressions by DAGs, p. 25.

91/17	A.T.M. Aerts P.M.E. de Bra K.M. van Hee	Transforming Functional Database Schemes to Relational Representations, p. 21.
91/18	Rik van Geldrop	Transformational Query Solving, p. 35.
91/19	Erik Poll	Some categorical properties for a model for second order lambda calculus with subtyping, p. 21.
91/20	A.E. Eiben R.V. Schuwer	Knowledge Base Systems, a Formal Model, p. 21.
91/21	J. Coenen W.-P. de Roever J.Zwiers	Assertional Data Reification Proofs: Survey and Perspective, p. 18.
91/22	G. Wolf	Schedule Management: an Object Oriented Approach, p. 26.
91/23	K.M. van Hee L.J. Somers M. Voorhoeve	Z and high level Petri nets, p. 16.
91/24	A.T.M. Aerts D. de Reus	Formal semantics for BRM with examples, p. 25.
91/25	P. Zhou J. Hooman R. Kuiper	A compositional proof system for real-time systems based on explicit clock temporal logic: soundness and complete ness, p. 52.
91/26	P. de Bra G.J. Houben J. Paredaens	The GOOD based hypertext reference model, p. 12.
91/27	F. de Boer C. Palamidessi	Embedding as a tool for language comparison: On the CSP hierarchy, p. 17.
91/28	F. de Boer	A compositional proof system for dynamic proces creation, p. 24.
91/29	H. Ten Eikelder R. van Geldrop	Correctness of Acceptor Schemes for Regular Languages, p. 31.
91/30	J.C.M. Baeten F.W. Vaandrager	An Algebra for Process Creation, p. 29.
91/31	H. ten Eikelder	Some algorithms to decide the equivalence of recursive types, p. 26.
91/32	P. Struik	Techniques for designing efficient parallel programs, p. 14.
91/33	W. v.d. Aalst	The modelling and analysis of queueing systems with QNM-ExSpect, p. 23.
91/34	J. Coenen	Specifying fault tolerant programs in deontic logic, p. 15.

91/35	F.S. de Boer J.W. Klop C. Palamidessi	Asynchronous communication in process algebra, p. 20.
92/01	J. Coenen J. Zwiers W.-P. de Roever	A note on compositional refinement, p. 27.
92/02	J. Coenen J. Hooman	A compositional semantics for fault tolerant real-time systems, p. 18.
92/03	J.C.M. Baeten J.A. Bergstra	Real space process algebra, p. 42.
92/04	J.P.H.W.v.d.Eijnde	Program derivation in acyclic graphs and related problems, p. 90.
92/05	J.P.H.W.v.d.Eijnde	Conservative fixpoint functions on a graph, p. 25.
92/06	J.C.M. Baeten J.A. Bergstra	Discrete time process algebra, p.45.
92/07	R.P. Nederpelt	The fine-structure of lambda calculus, p. 110.
92/08	R.P. Nederpelt F. Kamareddine	On stepwise explicit substitution, p. 30.
92/09	R.C. Backhouse	Calculating the Warshall/Floyd path algorithm, p. 14.
92/10	P.M.P. Rambags	Composition and decomposition in a CPN model, p. 55.
92/11	R.C. Backhouse J.S.C.P.v.d.Woude	Demonic operators and monotype factors, p. 29.
92/12	F. Kamareddine	Set theory and nominalisation, Part I, p.26.
92/13	F. Kamareddine	Set theory and nominalisation, Part II, p.22.
92/14	J.C.M. Baeten	The total order assumption, p. 10.
92/15	F. Kamareddine	A system at the cross-roads of functional and logic programming, p.36.
92/16	R.R. Seljée	Integrity checking in deductive databases; an exposition, p.32.
92/17	W.M.P. van der Aalst	Interval timed coloured Petri nets and their analysis, p. 20.
92/18	R.Nederpelt F. Kamareddine	A unified approach to Type Theory through a refined lambda-calculus, p. 30.
92/19	J.C.M.Baeten J.A.Bergstra S.A.Smolka	Axiomatizing Probabilistic Processes: ACP with Generative Probabilities, p. 36.
92/20	F.Kamareddine	Are Types for Natural Language? P. 32.

92/21	F.Kamareddine	Non well-foundedness and type freeness can unify the interpretation of functional application, p. 16.
92/22	R. Nederpelt F.Kamareddine	A useful lambda notation, p. 17.
92/23	F.Kamareddine E.Klein	Nominalization, Predication and Type Containment, p. 40.
92/24	M.Codish D.Dams Eyal Yardeni	Bottum-up Abstract Interpretation of Logic Programs, p. 33.
92/25	E.Poll	A Programming Logic for Fω, p. 15.
92/26	T.H.W.Beelen W.J.J.Stut P.A.C.Verkoulen	A modelling method using MOVIE and SimCon/ExSpect, p. 15.
92/27	B. Watson G. Zwaan	A taxonomy of keyword pattern matching algorithms, p. 50.
93/01	R. van Geldrop	Deriving the Aho-Corasick algorithms: a case study into the synergy of programming methods, p. 36.
93/02	T. Verhoeff	A continuous version of the Prisoner's Dilemma, p. 17
93/03	T. Verhoeff	Quicksort for linked lists, p. 8.
93/04	E.H.L. Aarts J.H.M. Korst P.J. Zwietering	Deterministic and randomized local search, p. 78.
93/05	J.C.M. Baeten C. Verhoef	A congruence theorem for structured operational semantics with predicates, p. 18.
93/06	J.P. Veltkamp	On the unavoidability of metastable behaviour, p. 29
93/07	P.D. Moerland	Exercises in Multiprogramming, p. 97
93/08	J. Verhoosel	A Formal Deterministic Scheduling Model for Hard Real-Time Executions in DEDOS, p. 32.
93/09	K.M. van Hee	Systems Engineering: a Formal Approach Part I: System Concepts, p. 72.
93/10	K.M. van Hee	Systems Engineering: a Formal Approach Part II: Frameworks, p. 44.
93/11	K.M. van Hee	Systems Engineering: a Formal Approach Part III: Modeling Methods, p. 101.
93/12	K.M. van Hee	Systems Engineering: a Formal Approach Part IV: Analysis Methods, p. 63.
93/13	K.M. van Hee	Systems Engineering: a Formal Approach

		Part V: Specification Language, p. 89.
93/14	J.C.M. Baeten J.A. Bergstra	On Sequential Composition, Action Prefixes and Process Prefix, p. 21.
93/15	J.C.M. Baeten J.A. Bergstra R.N. Bol	A Real-Time Process Logic, p. 31.
93/16	H. Schepers J. Hooman	A Trace-Based Compositional Proof Theory for Fault Tolerant Distributed Systems, p. 27
93/17	D. Alstein P. van der Stok	Hard Real-Time Reliable Multicast in the DEDOS system, p. 19.
93/18	C. Verhoef	A congruence theorem for structured operational semantics with predicates and negative premises, p. 22.
93/19	G-J. Houben	The Design of an Online Help Facility for ExSpect, p.21.
93/20	F.S. de Boer	A Process Algebra of Concurrent Constraint Programming, p. 15.
93/21	M. Codish D. Dams G. Filé M. Bruynooghe	Freeness Analysis for Logic Programs - And Correctness?, p. 24.
93/22	E. Poll	A Typechecker for Bijective Pure Type Systems, p. 28.
93/23	E. de Kogel	Relational Algebra and Equational Proofs, p. 23.
93/24	E. Poll and Paula Severi	Pure Type Systems with Definitions, p. 38.
93/25	H. Schepers and R. Gerth	A Compositional Proof Theory for Fault Tolerant Real-Time Distributed Systems, p. 31.
93/26	W.M.P. van der Aalst	Multi-dimensional Petri nets, p. 25.
93/27	T. Kloks and D. Kratsch	Finding all minimal separators of a graph, p. 11.
93/28	F. Kamareddine and R. Nederpelt	A Semantics for a fine λ-calculus with de Bruijn indices, p. 49.
93/29	R. Post and P. De Bra	GOLD, a Graph Oriented Language for Databases, p. 42.
93/30	J. Deogun T. Kloks D. Kratsch H. Müller	On Vertex Ranking for Permutation and Other Graphs, p. 11.
93/31	W. Körver	Derivation of delay insensitive and speed independent CMOS circuits, using directed commands and production rule sets, p. 40.
93/32	H. ten Eikelder and H. van Geldrop	On the Correctness of some Algorithms to generate Finite Automata for Regular Expressions, p. 17.

93/33	L. Loyens and J. Moonen	ILIAS, a sequential language for parallel matrix computations, p. 20.
93/34	J.C.M. Baeten and J.A. Bergstra	Real Time Process Algebra with Infinitesimals, p.39.
93/35	W. Ferrer and P. Severi	Abstract Reduction and Topology, p. 28.
93/36	J.C.M. Baeten and J.A. Bergstra	Non Interleaving Process Algebra, p. 17.
93/37	J. Brunekreef J-P. Katoen R. Koymans S. Mauw	Design and Analysis of Dynamic Leader Election Protocols in Broadcast Networks, p. 73.
93/38	C. Verhoef	A general conservative extension theorem in process algebra, p. 17.
93/39	W.P.M. Nuijten E.H.L. Aarts D.A.A. van Erp Taalman Kip K.M. van Hee	Job Shop Scheduling by Constraint Satisfaction, p. 22.
93/40	P.D.V. van der Stok M.M.M.P.J. Claessen D. Alstein	A Hierarchical Membership Protocol for Synchronous Distributed Systems, p. 43.
93/41	A. Bijlsma	Temporal operators viewed as predicate transformers, p. 11.
93/42	P.M.P. Rambags	Automatic Verification of Regular Protocols in P/T Nets, p. 23.
93/43	B.W. Watson	A taxomomy of finite automata construction algorithms, p. 87.
93/44	B.W. Watson	A taxonomy of finite automata minimization algorithms, p. 23.
93/45	E.J. Luit J.M.M. Martin	A precise clock synchronization protocol,p.
93/46	T. Kloks D. Kratsch J. Spinrad	Treewidth and Patwidth of Cocomparability graphs of Bounded Dimension, p. 14.
93/47	W. v.d. Aalst P. De Bra G.J. Houben Y. Kornatzky	Browsing Semantics in the "Tower" Model, p. 19.
93/48	R. Gerth	Verifying Sequentially Consistent Memory using Interface Refinement, p. 20.

94/01	P. America M. van der Kammen R.P. Nederpelt O.S. van Roosmalen H.C.M. de Swart	The object-oriented paradigm, p. 28.
94/02	F. Kamareddine R.P. Nederpelt	Canonical typing and Π-conversion, p. 51.
94/03	L.B. Hartman K.M. van Hee	Application of Marcov Decision Processe to Search Problems, p. 21.
94/04	J.C.M. Baeten J.A. Bergstra	Graph Isomorphism Models for Non Interleaving Process Algebra, p. 18.
94/05	P. Zhou J. Hooman	Formal Specification and Compositional Verification of an Atomic Broadcast Protocol, p. 22.
94/06	T. Basten T. Kunz J. Black M. Coffin D. Taylor	Time and the Order of Abstract Events in Distributed Computations, p. 29.
94/07	K.R. Apt R. Bol	Logic Programming and Negation: A Survey, p. 62.
94/08	O.S. van Roosmalen	A Hierarchical Diagrammatic Representation of Class Structure, p. 22.
94/09	J.C.M. Baeten J.A. Bergstra	Process Algebra with Partial Choice, p. 16.
94/10	T. verhoeff	The testing Paradigm Applied to Network Structure. p. 31.
94/11	J. Peleska C. Huizing C. Petersohn	A Comparison of Ward & Mellor's Transformation Schema with State- & Activitycharts, p. 30.
94/12	T. Kloks D. Kratsch H. Müller	Dominoes, p. 14.
94/13	R. Seljée	A New Method for Integrity Constraint checking in Deductive Databases, p. 34.
94/14	W. Peremans	Ups and Downs of Type Theory, p. 9.
94/15	R.J.M. Vaessens E.H.L. Aarts J.K. Lenstra	Job Shop Scheduling by Local Search, p. 21.
94/16	R.C. Backhouse H. Doornbos	Mathematical Induction Made Calculational, p. 36.
94/17	S. Mauw M.A. Reniers	An Algebraic Semantics of Basic Message Sequence Charts, p. 9.

94/18	F. Kamareddine R. Nederpelt	Refining Reduction in the Lambda Calculus, p. 15.
94/19	B.W. Watson	The performance of single-keyword and multiple-keyword pattern matching algorithms, p. 46.
94/20	R. Bloo F. Kamareddine R. Nederpelt	Beyond β-Reduction in Church's $\lambda\rightarrow$, p. 22.
94/21	B.W. Watson	An introduction to the Fire engine: A C++ toolkit for Finite automata and Regular Expressions.
94/22	B.W. Watson	The design and implementation of the FIRE engine: A C++ toolkit for Finite automata and regular Expressions.
94/23	S. Mauw and M.A. Reniers	An algebraic semantics of Message Sequence Charts, p. 43.
94/24	D. Dams O. Grumberg R. Gerth	Abstract Interpretation of Reactive Systems: Abstractions Preserving $\forall$CTL*, $\exists$CTL* and CTL*, p. 28.
94/25	T. Kloks	$K_{1,3}$-free and W_4-free graphs, p. 10.
94/26	R.R. Hoogerwoord	On the foundations of functional programming: a programmer's point of view, p. 54.
94/27	S. Mauw and H. Mulder	Regularity of BPA-Systems is Decidable, p. 14.