Orthogonal Polyhedra as Geometric Bounds in Constructive Solid Geometry

A. Aguilera
D. Ayala

Report LSI-96-64-R

ORTHOGONAL POLYHEDRA AS GEOMETRIC BOUNDS IN CONSTRUCTIVE SOLID GEOMETRY

A. Aguilera and D. Ayala
Universitat Politècnica de Catalunya
Barcelona, Spain
ayala@lsi.upc.es, aguilera@goliat.upc.es

November 28, 1996

Abstract

Set membership classification and, specifically, the evaluation of a CSG tree are problems of a certain complexity. Several techniques to speed up these processes have been proposed such as Active Zones, Geometric Bounds and the Extended Convex Differences Tree.

Boxes are the most common geometric bounds studied but other bounds such as spheres, convex hulls and prisms have also been proposed.

On the other hand, there is an extended bibliography dealing with convex polyhedra and solving problems for this class of polyhedra. Orthogonal polyhedra are also a class of polyhedra and several problems have been solved for them.

In this work we propose orthogonal polyhedra as geometric bounds in the CSG model. CSG primitives are approximated by orthogonal polyhedra and the orthogonal bound of the object is obtained by applying the corresponding boolean algebra. A specific model for orthogonal polyhedra is presented that allows a simple and robust boolean operations algorithm between orthogonal polyhedra. This algorithm has linear complexity (is based on a merging process) and avoids floating-point computation.

1 Introduction

Constructive solid geometry is a non-ambiguous 3D model that allows to build up complicated shapes from simple ones. This model is represented by a tree in which internal nodes represent boolean regularized operations and the leaf nodes represent simple shapes or primitives [15].

Set membership classification [20] and, specifically, the boundary evaluation of a CSG tree are problems of a certain complexity. Until now, several accelerating techniques have been proposed to speed up geometric computations in CSG. Among them we emphasize on Active Zones [17], [4] the Extended Convex Differences Tree [14] and Approximating Shapes or Geometric Bounds [8], [5].

The more extensively used geometric bounds are the well-known bounding boxes, but other shapes as spheres and convex hulls have also been proposed and studied [5].

On the other hand, in several disciplines such as solid modeling and computational geometry, it is very usual to start studying problems on simpler classes of polyhedra rather than on the general case. The most usually chosen is the convex polyhedra class. Convexity enables the use of efficient and simple algorithms [13], [6]. Orthogonal polyhedra are a less used simple class. Nevertheless, some works have been published dealing with this simpler class [10], [9], [3]. The restricted class of convex and orthogonal polyhedra, i.e., orthogonal boxes have been widely used in many applications [13], [8], [18].

In this work we propose orthogonal polyhedra as geometric bounds in CSG. We define a specific model to represent such class of polyhedra, the Extreme Vertices (EV) model. Then, in order to compute the orthogonal bound for a CSG object, we have developed a robust algorithm for regularized boolean operations. This algorithm has linear complexity (is based on a merging process) and avoids floating- point computations.

The paper is arranged as follows. Sections 2 and 3 deal respectively on geometric bounds and on orthogonal polyhedra, analyzing the related work on both disciplines. Section 4 defines the Extreme Vertices, EV, model and section 5 describes the corresponding boolean operations algorithm. Section 6 discusses the advantages and drawbacks of using orthogonal polyhedra instead of classical boxes. Finally, section 7 summarizes the conclusions and also shows possible directions for future work.

2 Geometric Bounds

A common way to reduce the complexity of geometric computations in CSG is the use of geometric bounds or approximating shapes. After fixing a class $\sum$ of approximating shapes, the process to be done consists on [8]:

1. All the primitives p in the tree are approximated with their corresponding approximating shape, $p \rightarrow AS(p) \epsilon \sum$
2. A postorder tree traversal is done by applying the following rules:
   (a) if $T = T1 \cup T2 \rightarrow AS(T) = AS(AS(T1) \cup AS(T2)) = AS(T1) \sqcup AS(T2)$
   (b) if $T = T1 \cap T2 \rightarrow AS(T) = AS(AS(T1) \cap AS(T2)) = AS(T1) \sqcap AS(T2)$
   (c) if $T = T1 - T2 \rightarrow AS(T) = AS(T1)$

Hence, the approximating shapes for all the internal nodes and for the root representing the object are determined.

The symbols $\sqcup$ and $\sqcap$ refer to operators equivalent to the boolean operations but closed into the $\sum$ class.

In [4] the S-bound theory is introduced and in [5] is formally developed. A class of totally consistent bounding functions is defined and the initial principle working with geometric bounds is extended by the application of the so called upward and downward rules:

upward rule : is the above mentioned postorder tree traversal

downward rule : the geometric bound of each node is refined by intersecting it with the geometric bound of its father,

$$AS(T) = AS(T) \sqcap AS(T.father)$$

Both rules are continuously applied until convergence is reached. For a detailed discussion concerning S-bounds, see [5].

Boxes have been the more widely used geometric bounds. A box is defined as:

$$A = < x_{Am}, y_{Am}, z_{Am}, x_{AM}, y_{AM}, z_{AM} >= \\ \{(x, y, z) | x_{Am} \leq x \leq x_{AM}, y_{Am} \leq y \leq y_{AM}, z_{Am} \leq z \leq z_{AM}\}$$

And the corresponding operators are defined as [11]:

$C = A \sqcup B$

where,

$$\begin{array}{ll} x_{Cm} = min(x_{Am}, x_{Bm}) & x_{CM} = max(x_{AM}, x_{BM}) \\ y_{Cm} = min(y_{Am}, y_{Bm}) & y_{CM} = max(y_{AM}, y_{BM}) \\ z_{Cm} = min(z_{Am}, z_{Bm}) & z_{CM} = max(z_{AM}, z_{BM}) \end{array}$$

and

$C = A \sqcap B$

where

$$\begin{array}{ll} x_{Cm} = max(x_{Am}, x_{Bm}) & x_{CM} = min(x_{AM}, x_{BM}) \\ y_{Cm} = max(y_{Am}, y_{Bm}) & y_{CM} = min(y_{AM}, y_{BM}) \\ z_{Cm} = max(z_{Am}, z_{Bm}) & z_{CM} = min(z_{AM}, z_{BM}) \end{array}$$

We can easily observe that while the operator $\sqcap$ coincides with the intersection, the $\sqcup$ operator does not correspond to the union operation. The operators $\sqcup$ and $\sqcap$ over the class bounding boxes are a non-distributive lattice instead of a boolean algebra [11].

An order relation can be defined in a lattice such as:

$$a \preceq b \Leftrightarrow a \cap b = a$$

and, from lattice theory, the following distributive inequalities are obtained:

$$(a \cup b) \cap c \succeq (a \cap c) \cup (b \cap c)$$

$$(a \cap b) \cup c \preceq (a \cup c) \cap (b \cup c)$$

Based on these inequalities, in [11] the authors show that the bounding box size of a CSG depends on the form of its algebraic expression and that the smallest bounding box is obtained when this algebraic expression is in the normal disjunctive form (DF) or union of intersections form (UOI). The authors also show that, in general, this technique produces better bounds than the S-bounds technique. In [7] an algorithm is presented that converts a CSG expression into its DF.

In [16] other advantages of DF are shown:

1. DF only requires a stack of depth 1 and then it has been used for evaluating CSG trees in parallel.
2. When CSG primitives are halfspaces, intersections are convex polyhedra and then the CSG object can be represented as the union of convex polyhedra.
3. We can avoid visiting all the primitives for all intersections. When the intersection currently visited contains a combination of primitives that resulted in empty bounds for a previously visited intersection, then we can state that this current intersection is empty without visiting its remaining terms. This fact is referred as culling up empty intersections.

Nevertheless, the size of the DF grows exponentially in the number of primitives of the original tree. So, in order to alleviate the need of storing such a large tree in [16] an algorithm is presented that processes the DF directly from the initial tree.

3 Orthogonal Polyhedra

Orthogonal polyhedra (OP) are polyhedra with all their faces oriented in three orthogonal directions. In this work we will consider only two-manifold OP.

This class of polyhedra implies a restriction of the general case concerning with the geometry. In an OP, all planes and lines are parallel to three orthogonal axes, the number of incident edges for any vertex can be only three, four or six [9] and faces have an even number of edges (vertices). These geometric characteristics make OP be a more restricted class than convex polyhedra. However, concerning with the topology, OP do not imply any restriction at all. OP allow any number of rings on faces, holes (they can be of any genus) and shells. Then, they represent a radically different class of polyhedra than the convex class represents.

There is a large amount of work concerning with convex polyhedra but its study is not the purpose of the present work.

OP are a less used simple class though some studies have been published dealing with or using them. In [10] a B-Rep to CSG conversion algorithm is presented that works for a restricted class of OP. The obtained CSG expression is a Peterson-style formula and the restricted class are the acyclic OP. In [9] the same author extends the domain for a certain class of cyclic OP. In [12] an octree to B-Rep conversion algorithm is presented and an OP is obtained. In [3] an algorithm that simplifies geometry is presented for the particular case of OP; a more complex algorithm is needed for the general case of polyhedra [2].

Boxes, which are both convex and orthogonal, have been widely used in many applications [13], [8], [18] and have been used as approximations as has been explained in the previous section.

4 Extreme Vertices Model for Orthogonal Polyhedra

In this section we present a model for two-manifold OP. We consider that all the OP, as well as their geometric elements (faces and edges) with which we operate, are in the same iso-oriented coordinate system.

The Extreme Vertices model, EV, represents OP in a complete and compact way. The model is complete because we can infer from it all the topological and geometric information of the polyhedron.

Splitting and boolean set operations can be done on EV in linear time. Although input data (i.e., coordinates vertices) are floating-point values, no time-consuming floating-point arithmetic is ever performed, so there are no propagation errors. All results are obtained by just classifying vertices coordinates of the initial data.

Other operations such as computing the perimeter, area and volume of OP as well as conversion algorithms between EV and hierarchical B-Rep, Classical Octrees and Extended Octrees have also been developed [1].

As mentioned in the previous section, in an OP the number of edges incident on a vertex can be 3, 4 or 6. From now on we will refer to them as V3, V4 or V6.

Definition 4.1 *A* brink *is the longest uninterrupted segment, built out of a sequence of contiguous collinear edges of an OP.*

Every edge belongs to a brink, whereas every brink consists of one or more edges and contains as many vertices as the number of edges plus one (see figure 1 right).

Edges meeting at a V3 vertex are all linearly independent whereas edges meeting at V4 or V6 vertices are not. Edges meeting at a V4 (V6) vertex belong to two (three) perpendicular directions, that is, they are members of two (three) perpendicular brinks and, hence, they appear as two (three) couples of collinear edges (see figure 1 left). Also every V4 or V6 incident edge has a neighbour in the brink corresponding to its direction.

Lemma 4.1 *In a brink both ending vertices are V3 and the remaining (interior) are V4 or V6.*

Proof: Every edge meeting at vertices V4 or V6 has a neighbour in the same brink, then such vertices cannot appear at the end of any brink. Moreover, any edge meeting at vertices V3 has not a neighbour in the same brink and therefore such vertices must appear only at the end of any brink.

Definition 4.2 *We will call* Extreme Vertices *(EV) of an OP to the ending vertices of all the OP brinks, i.e. the V3 vertices of the polyhedron.*

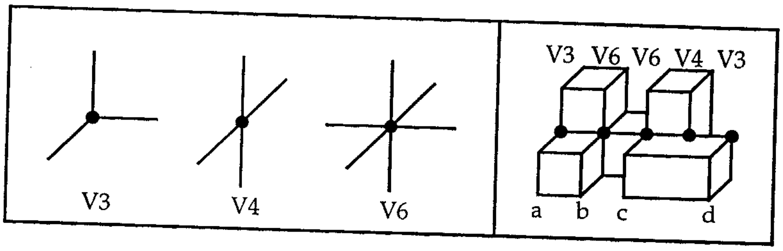


Figure 1: Left) Edges meeting at a V3, V4 and V6 vertex. Right) Example of a brink containing four edges and five vertices. These vertices are respectively V3, V6, V6, V4 and V3. Edges (a, b) and (c, d) are collinear but are not contiguous and then (a, b) is a brink and (c, d) is another brink.

Definition 4.3 *We define the* EV model *for OP as a model that only stores all EV (V3) vertices.*

Lemma 4.2 *Let P be an OP and OH(P) be its isooriented orthogonal hull or minimum bounding box. Then, only a subset of V3 vertices of P lies on the boundary of OH(OP) and, therefore, all V4 and V6 vertices lie in the interior of OH(P).*

Proof: The proof comes from the well-known concept of supportability [19]. Concerning with OP, V3 vertices are locally or complementary supportable and vertices V4 and V6 are non-supportable [9]. Only supportable vertices of an OP can lie on its minimum bounding box.

Lemma 4.3 *Let* $VX = \{x_1, x_2, \ldots, x_{nx}\}$ *be the ordered set of different values for the* x *coordinate of every V3 vertex,* nx *being the total number of different* x *values (and* VY *and* VZ *analogously for their* y *and* z *coordinates, with sizes* ny *and* nz*).*

Then, for every vertex V4 or V6 with coordinates (x_i, y_i, z_i), $x_i \in \{x_2, x_3, \ldots, x_{nx-1}\} = VX - \{x_1, x_{nx}\}$ *(and analogously for its* y *and* z *coordinates).*

Proof: Vertices V4 (V6) are in the interior of 2 (3) perpendicular brinks (they are indeed the intersection of these brinks), so their coordinates can be obtained from the coordinates of the V3 ending vertices of these brinks, then $x_i \in VX$. However, from lemma 4.2, $x_1 < x_i < x_{nx}$, and therefore V4 and V6 are in the interior of the bounding box of their OP.

Theorem 4.1 *The EV model for orthogonal polyhedra is a valid B-Rep model.*

Proof: From lemma 4.3, all coordinates of vertices V4 and V6 appear as coordinates of vertices V3. Then, although vertices V4 and V6 do not appear in the model, they can be inferred from it.

See [1] for the conversion algorithm from EV to a hierarchical B-rep model.

5 Boolean Operations in the EV model

Our approach computes an orthogonal bound for a CSG tree. We consider CSG trees without geometric transformations.

First, all the primitives are approximated by their bounding boxes and then a postorder tree traversal is done applying the corresponding operations. In our case the operations are the classical operations of the boolean algebra and so the orthogonal bound of the CSG does not depend on the form of the CSG algebraic expression as occurred with boxes (see section 2).

Nevertheless, the advantage of the DF form concerning with culling up empty intersections, also mentioned in section 2, can be applied to our approach and, therefore, the tree traversal is done by using the method proposed in [16]. Furthermore, being boxes the CSG primitives, intersections are also boxes (when they are not empty) and so the CSG bound is represented as the union of boxes.

In this section the boolean operations algorithm for OP is presented. The algorithm basically performs a geometric merge between OP represented in a sorted EV model. The algorithm computes a sequence of 2D sections from the 3D model and the same algorithm is recursively applied to each of these 2D sections obtaining 1D sections. Then 1D boolean operations are performed on these 1D sections. The recursion upwards by converting the resulting sequence of sections into an EV model thus obtaining the corresponding result.

5.1 Operations on the EV model

Definition 5.1 *An* ABC-sorted EV model *is an EV model where vertices V3 are sorted first by coordinate A, then by B and then by C.*

EV models can be sorted on six different ways: XYZ, XZY, YXZ, YZX, ZXY and ZYX. In a ZXY-sorted EV model, for instance, its vertices are arranged in planes perpendicular to the Z axis (i.e. with the same z coordinate). In each such a plane they are arranged in lines parallel to the Y axis (i. e. with the same x coordinate). Finally they appear as y intervals (see figure 2).

Let us have an ABC-sorted EV model,

Definition 5.2 *A* plane of vertices *of an OP is the set of vertices lying on a plane perpendicular to the A axis. We will also refer as* line of vertices *to the set of vertices lying on a line parallel to the C axis within a plane of vertices.*

Definition 5.3 *A* strip *is the region between two consecutive planes (lines) of vertices.*

Definition 5.4 *A* section *is the polygon resulting from the intersection between an OP and an orthogonal plane perpendicular to the A axis which must not coincide with any plane of vertices.*

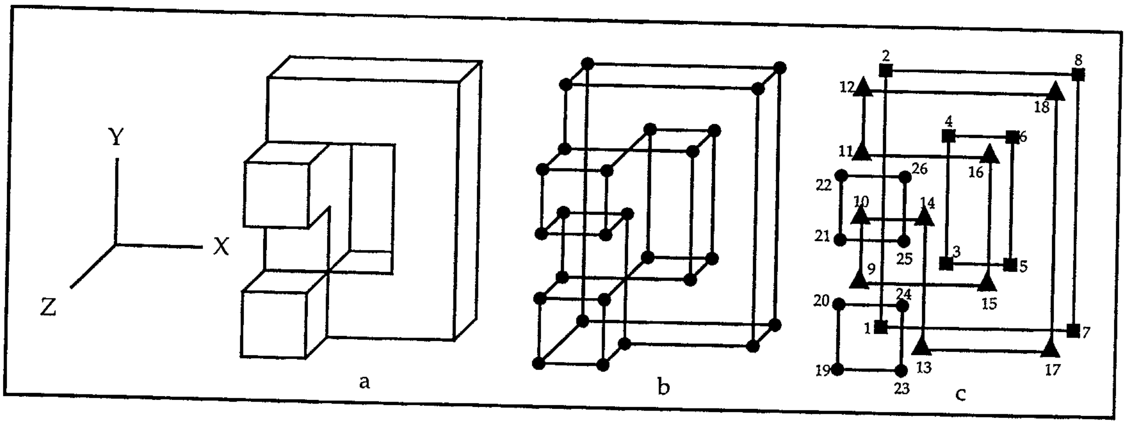


Figure 2: ZXY-sorted EV model. a) A hidden line representation of an OP with one V6, one V4 and 26 V3s. b) Its corresponding wire-frame representation. c) This representation shows the order number for each V3 vertex and the three planes of vertices of the model (with different marks)

All the orthogonal planes intersecting an OP in the same strip give the same section. Hence, every strip has its representing section. Furthermore, as an OP can be interpreted as a sequence of strips, we can define the sequence of sections for an OP.

All these concepts related to sections can be defined also in 2D. A section is also a 1D polygon resulting from the intersection between a 2D orthogonal polygon and an orthogonal line perpendicular to both A and B axes which must not coincide with any line of vertices.

A sorted-EV model is a sequence of planes (lines) of vertices. The number of elements of this sequence, np, is the number of different A coordinates in the model. The number of sections is $ns = np + 1$ because the empty sections S_0 and S_{np} are also considered. Figure 3 shows the sections and planes of vertices for an OP.

An ABC-sorted EV model can represent n-dimensional OP ($n \leq 3$) by taking into account the last n coordinates. Then planes and lines of vertices of an OP will be represented also in this model. Moreover, as a section is actually an OP, 1D and 2D sections will also be represented in this model.

Then, we define the *ABCsorted* type with the following operations:

```
FUNCTION IniEv () RETURN ABCsorted
{Returns an empty EV model}

PROCEDURE Put (INPUT plv: ABCsorted, I/O P: ABCsorted, INPUT dim:INTEGER)
{Appends a plane (dim=2) or a line (dim=1) to an EV model}

FUNCTION Read (P: ABCsorted, dim:INTEGER) RETURN ABCsorted
{Extracts the next plane (dim=2) or line (dim=1) from an EV model}

FUNCTION End (P: ABCsorted) RETURN BOOLEAN
{Returns TRUE if the end of P has been reached}
```

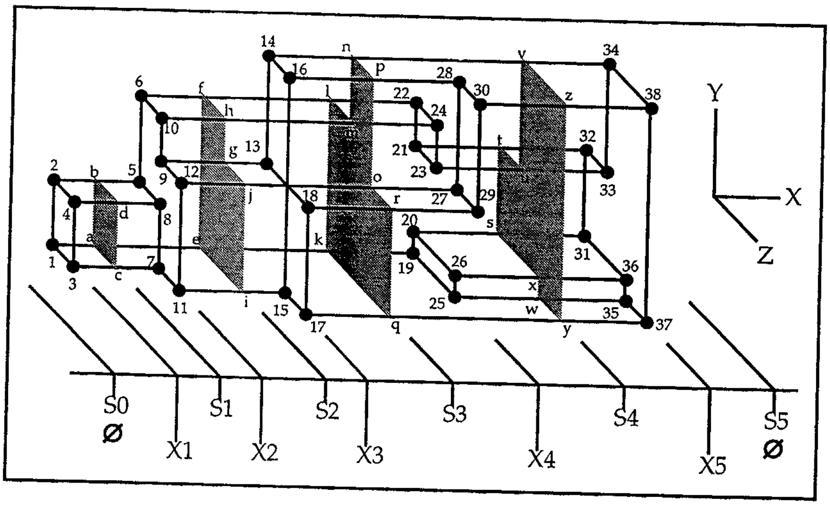


Figure 3: This OP is represented in an XZY-sorted EV model. It has 5 planes of vertices, X_1 to X_5. Each of them corresponds to the set of vertices with the same coordinate X. The shadowed polygons are the four sections S_1, S_2, S_3 and S_4 and there are two more empty sections, S_0 and S_5.

```
FUNCTION MergeXor (P, Q: ABCsorted; dim: INTEGER) RETURN ABCsorted
{Applies the Exclusive OR operation to the vertices of P and Q and
returns the resulting set}

PROCEDURE SetCoord (I/O P: ABCsorted, INPUT Coord: REAL,
                    INPUT dim: INTEGER)
{Sets the A (dim=2) or the B (dim=1) coordinate to Coord on every
vertex of the plane (line) of vertices P}

FUNCTION GetCoord (P: ABCsorted, dim:INTEGER) RETURN REAL
{Gets the common A (dim=2) or B (dim=1) coordinate of the plane (line)
of vertices P}
```

5.2 Computing sections from planes (lines) of vertices and vice versa

Any section S_i is computed by doing an exclusive OR between its previous section S_{i-1} and its previous plane (line) of vertices P_i:

$$S_i = S_{i-1} \otimes P_i, \forall i \in [1, np-1]$$

and

$$S_0 = \emptyset, S_{np} = \emptyset,$$

$\otimes$ means the exclusive OR operation.

Then we define the corresponding function GetSection:

```
FUNCTION GetSection (S: ABCsorted, plv: ABCsorted, dim: INTEGER) RETURN
ABCsorted
{returns the next section of an OP whose previous section is S. This
function works
for dimension 2 or 1. If dim=2 (dim=1), plv is the previous plane (line)
of vertices and S is a 2D (1D) section}

    RETURN (MergeXor(S, plv, dim))
ENDFUNCTION
```

An algorithm that computes the sequence of sections of an OP from its EV model using functions IniEv and GetSection is presented in [1].

A plane (line) of vertices P_i of an OP is computed by doing an exclusive OR between its previous S_{i-1} and next S_i sections:

$$P_i = S_{i-1} \otimes S_i, \forall i \in [1, np]$$

Then we define the corresponding function GetPlv:

```
FUNCTION GetPlv (Si: ABCsorted, Sj: ABCsorted, dim: INTEGER) RETURN
ABCsorted
{This function also works for dimensions 2 or 1.
If dim=2 (dim=1), Si and Sj are 2D (1D) consecutive sections and
returns the plane (line) of vertices between Si and Sj.}

    RETURN (MergeXor (Si, Sj, dim)
ENDFUNCTION
```

Actually, this function performs the same computations that the GetSection function, i.e., an exclusive OR between two sets of vertices, but as they are conceptually different we will use both of them.

An algorithm that computes the EV model from a sequence of sections of an OP is also presented in [1].

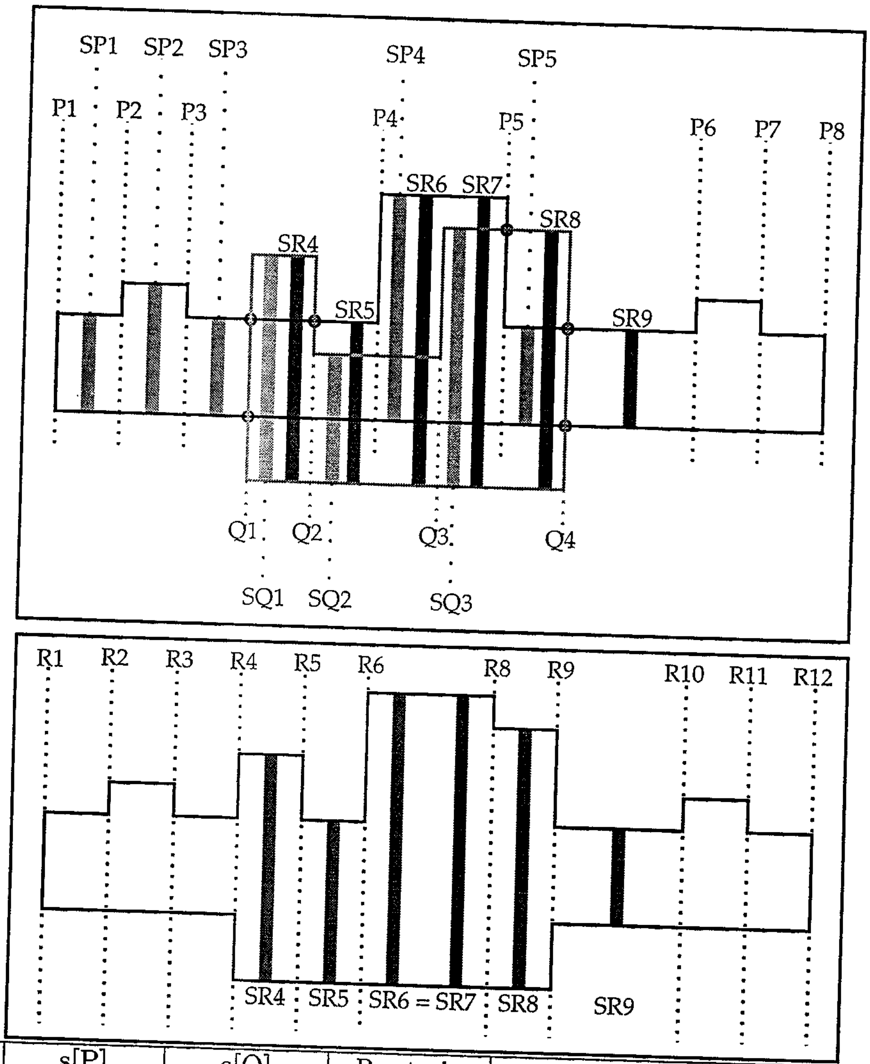
5.3 Boolean Operations algorithm

Now, we are able to present the boolean operations algorithm. The algorithm merges two OP, say P and Q, represented in the same ABC-sorted EV model, in such a way that the corresponding planes of vertices become also merged. We consider all the resulting strips. Some of them will correspond to untouched strips of P or Q and only one section will have to be considered. However some other strips will correspond to a part of a P strip and a part of a Q strip with their corresponding sections. The algorithm considers this sections as 2D OP and operates them in the same way.

We can explain the algorithm as follows. The sequence of sections for objects P and Q are computed. Then these sections are merged in order to compute the sequence of sections of the R resulting object. Finally, from this sequence of sections, the EV model of the resulting object R is obtained. Nevertheless, the implemented algorithm does not work in this sequential form; it actually works in a wholly merged form and only needs to store one section for each of the P and Q operands and two consecutive sections for the result R. Then, the algorithm is O(n) as merging-like algorithms are.

```
TYPE Object = ENUM {P, Q} ENDTYPE
FUNCTION OpBool (P, Q: ABCsorted,  {the input objects}
                 dim: INTEGER,     {dimension of P and Q}
                 op: BoolOp)       {the Boolean operation}
                 RETURN ABCsorted
  VAR
    s[P..Q]: ABCsorted  {s[P], s[Q]: current sections of P, Q}
    sRprevious, sRcurrent: ABCsorted  {sections of the result, R}
    plvi, plvo: ABCsorted  {input and output planes (lines) of vertices}
    obj: Object  {the current selected object}
    coord: REAL {The common coordinate of a plane (line) of vertices}
  ENDVAR
  IF dim = 1 THEN
     RETURN (OpBool1D(P, Q, op))
  ELSE
     dim:= dim - 1
     s[P]:= IniEv()
     s[Q]:= IniEv()
     sRcurrent:= IniEv()
     GetPlane(P, Q, dim, plvi, coord, obj)
     WHILE NOT End(P) AND NOT End(Q) DO
         S[obj]:= GetSection(plvi, S[obj], dim)
         sRprevious:= sRcurrent
         sRcurrent:= OpBool(s[P], s[Q], dim, op)
         plvo:= GetPlv(sRprevious, sRcurrent, dim)
         SetCoord(plvo, coord, dim)
         Put(plvo, R, dim)
         GetPlane(P, Q, dim, plvi, coord, obj)
     ENDWHILE
     WHILE NOT End(P)  DO
         PutBool(plvi, R, op); plvi:= Read(P, dim)
     ENDWHILE
     WHILE NOT End(Q)  DO
         PutBool(plvi, R, op); plvi:= Read(Q, dim)
     ENDWHILE
     RETURN (R)
  ENDIF
ENDFUNCTION
```

Function OpBool1D performs 1D boolean operations between P and Q that now are



plvi	s[P]	s[Q]	sRanterior	sRactual	plvo
P1	SP1 = P1	∅	∅	SP1	R1 = P1
P2	SP2	∅	SP1	SP2	R2 = P2
P3	SP3	∅	SP2	SP3	R3 = P3
Q1	SP3	SQ1 = Q1	SP3	SR4 = SP3 ∪ SQ1	R4
Q2	SP3	SQ2	SR4	SR5 = SP3 ∪ SQ2	R5
P4	SP4	SQ2	SR5	SR6 = SP4 ∪ SQ2	R6
Q3	SP4	SQ3	SR6	SR7 = SP4 ∪ SQ3	R7= ∅
P5	SP5	SQ3	SR7	SR8 = SP5 ∪ SQ3	R8
Q4	SP5	SQ4 = ∅	SR8	SR9 = SP5 ∪ SQ4	R9
P6					R10 = P6
P7					R11 = P7
P8					R12 = P8

Figure 4: Boolean Operations running example. End(Q)is TRUE when Q4 is selected. We can observe that SR6=SR7 since SP4 ∪ SQ2 = SP4 ∪ SQ3, thus making R7 = ∅

collinear lines of vertices.

Procedure GetPlane gets the next plane (line) of vertices `plvi` of P or Q, with its common coordinate `coord`, and to which of these objects `obj` it belongs. The plane (line) of vertices is obtained using function Read and its common coordinate using function GetCoord (see section 5.1). This procedure works as in a merging process.

Functions GetSection and GetPlv perform an exclusive OR between the sets of vertices of their operands (see subsection 5.2).

OpBool works for 3D OP (dim=3) and for 2D orthogonal polygons (dim=2). The recursive case of this procedure is a merging-like algorithm.

When the end of one of the objects is reached, the main iteration finishes and the remaining planes (lines) of vertices of the other object are either appended or not to the result object depending on the considered boolean operation. Procedure PutBool performs this boolean operation based appending process.

Figure 4 shows a 2D running example and figure 5 shows a 3D example.

6 Comparison between geometric bounds

In this work we propose the EV Model as a new model for representing valid OP in a compact way. This model allows performing robust Boolean operations in O(n) complexity.

We have also developed classification algorithms for OP (represented in the mentioned ABC-sorted model). In [1] an O(n) splitting algorithm and an O(lgn) point classification algorithm are proposed.

We want now to compare OP with simple boxes as geometric bounds. Boolean operations on boxes are of constant complexity whereas boolean operations on OP are O(n). Classification algorithms are also more complex for OP than for boxes (for boxes are of constant complexity). So, our approach will be more time consuming than boxes based approaches when the CSG geometric bound is computed and when classification tests are performed on it. Nevertheless, OP are tighter than boxes, therefore classification tests will be more deterministic.

Moreover, the bounding OP used for a primitive is just its bounding box and we will traverse the CSG tree in its DF using the method presented in [16] i.e. performing unions of intersections.. Then, when performing intersections our method deals also with boxes and has constant complexity. Obviously, the method must finally perform unions between the intersection results and then complexity is O(n).

7 Conclusions and Future work

In this work we have proposed the use of OP as geometric bounds in CSG. The proposal is based on the fact that a simple boolean operations algorithm can be applied for OP. This boolean operations algorithm is a merging-like algorithm and runs in O(n).

Although input data (i.e. coordinates vertices) are floating-point values, no time-

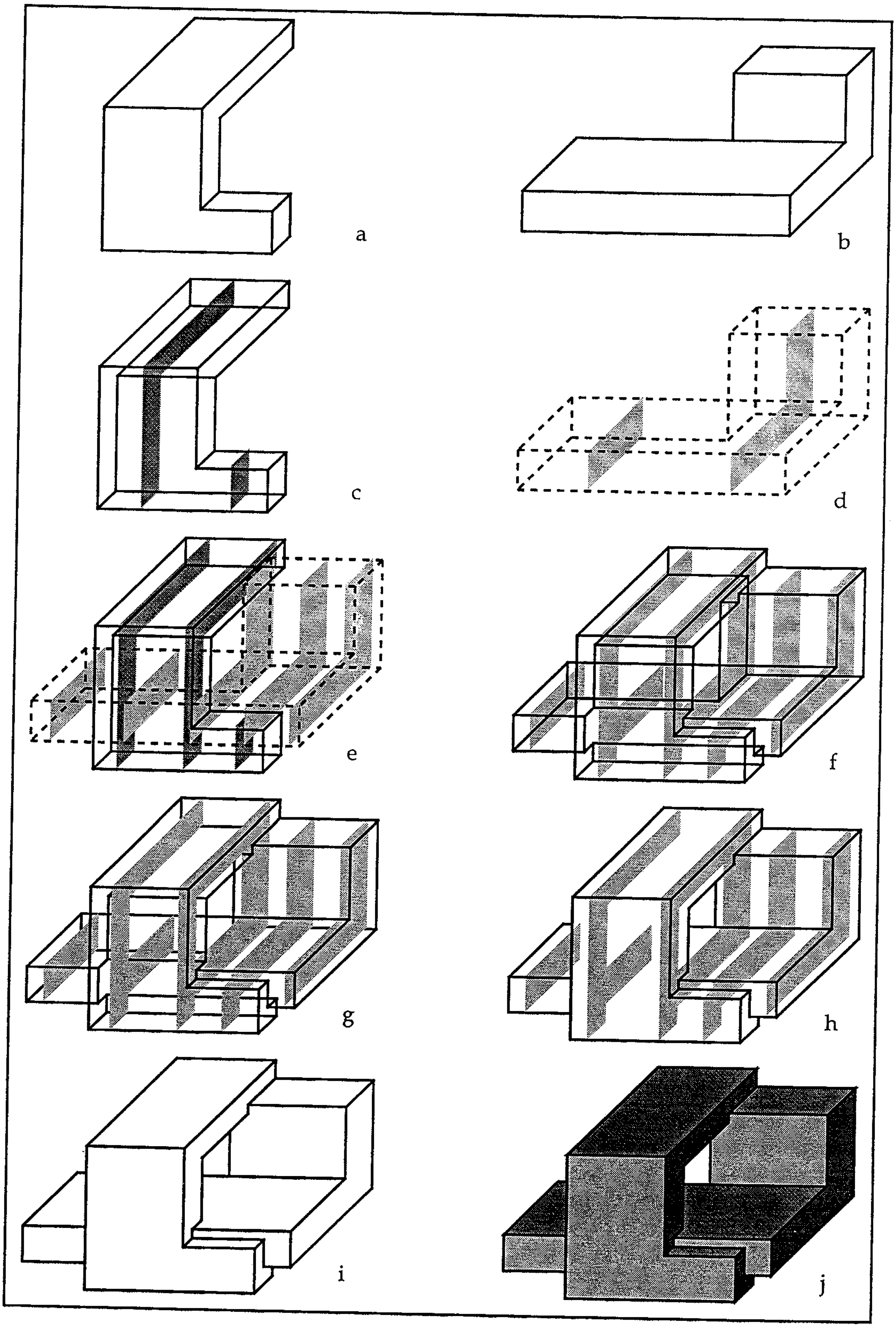


Figure 5: Boolean Operations: 3D example. (a),(b) Two OP. (c),(d) Sections of these OP. (e) OP in overlapping position and the corresponding overlapping sections. (f),(g),(h) The resulting sections and OP (wireframe and HLR). (i),(j) The resulting OP (HLR and shaded).

consuming floating-point arithmetic is ever performed, so there are no propagation errors. All results are obtained by just classifying vertex coordinates of the initial data. Moreover, round-off errors in the input data can be avoided by performing a space discretization based on the primitive bounding boxes.

Working with OP instead of boxes as geometric bounds is more time consuming when computing the bound and when classification algorithms are applied. However OP are tighter than boxes, therefore classification tests will be more deterministic.

As a future work we are intended to compare these theoretical results with experimental ones. We also are extending the EV model and the corresponding operations for non-manifold OP and, finally, we will study other applications of OP.

8 Acknowledgments

We thank P. Brunet and R. Joan-Arinyo for their valuable comments and suggestions.

References

[1] A. Aguilera and D. Ayala. The EV model for orthogonal polyhedra. Technical Report in preparation, LSI-Universitat Politecnica de Catalunya, 1996.

[2] C. Andújar, D. Ayala, P. Brunet, R. Joan-Arinyo, and J. Solé. Automatic generation of multiresolution boundary representations. volume 15. EUROGRAPHICS'96, 1996.

[3] D. Ayala, C. Andújar, and P. Brunet. Automatic simplification of orthogonal polyhedra. In D. Fellner, editor, *Modeling, Virtual Worlds, Distributed Graphics: Proceedings of the International MVD'96 Workshop.* Infix, 1995.

[4] S. Cameron and J.R. Rossignac. Relationship between S-bounds and Active Zones in Constructive Solid Geometry. Proceedings of the International Conference on the Theory and Practice of Geometric Modelling. FRG, 1988.

[5] S. Cameron and C. Yap. Refinement methods for geometric bounds in Constructive Solid Geometry. *ACM Transactions on Graphics*, 11(1):12 – 39, 1992.

[6] H. Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer Verlag, 1987.

[7] J. Goldfeather, S. Molnar, G. Turk, and H. Fuchs. Near real-time CSG rendering using tree normalization and geometric pruning. *IEEE Computer Graphics & Applications.*

[8] C. M. Hoffmann. *Geometric and Solid Modeling.* Morgan Kauffmann Publishers, Inc., 1989.

[9] R. Joan-Arinyo. Domain extension of isothetic polyhedra with minimal CSG representation. *Computer Graphics Forum*, (5):281 – 293, 1995.

[10] R. Juan-Arinyo. On Boundary to CSG and Extended Octrees to CSG conversions. In W. Strasser, editor, *Theory and Practice of Geometric Modeling*, pages 349–367. Springer-Verlag, 1989.

[11] M. Mazzetti and L. Ciminiera. Computing CSG-tree boundaries as algebraic expressions. *CAD*, 26(6):417 – 425, 1994.

[12] C. Montani and R. Scopigno. *Graphics Gems II*, chapter IV.7 Quadtree/Octree to boundary conversion, pages 202 – 218. Academic Press, Inc, 1991.

[13] F.P. Preparata and M.I. Shamos. *Computational Geometry: an Introduction.* Springer-Verlag, 1985.

[14] A. Rappoport. The n-dimensional Extended Convex Differences Tree (ECDT) for representing polyhedra. In J. Rossignac and J. Turner, editors, *Symposium on Solid Modeling Foundations and CAD/CAM Applications*, 1991.

[15] A. Requicha. Representations for rigid solids: Theory, methods, and systems. *Computing Surveys of the ACM*, 12:437–464, 1980.

[16] J. Rossignac. Proceesing disjunctive forms directly from CSG grafs. In *CSG 94. Set-theoretic Solid Modelling Techniques and Applications*, pages 55 – 70. Information Geometers Ltd, 1994.

[17] J. R. Rossignac and H. B. Voelcker. Active zones in CSG for accelerating boundary evaluation, redundancy elimination, interference detection and shading algorithms. *ACM Transactions on Graphics*, 8(1):51 – 87, 1989.

[18] H. Samet. *The Design and Analysis of Spatial Data Structures.* Addison Wesley Publ., Reading, MA, 1989.

[19] K. Tang and T. Woo. Algorithmic aspects of alternating sum of volumes. Part 2: Nonconvergence and its remedy. *CAD*, 23(6):435 – 443, 1991.

[20] R. B. Tilove. Set membership classification: a unified approach to geometric intersection problems. *IEEE Transactions on Computers*, 29(10):874 – 883, 1980.

Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya

Research Reports – 1996

LSI-96-1-R "(Pure) Logic out of Probability", Ton Sales.

LSI-96-2-R "Automatic Generation of Multiresolution Boundary Representations", C. Andújar, D. Ayala, P. Brunet, R. Joan-Arinyo, and J. Solé.

LSI-96-3-R "A Frame-Dependent Oracle for Linear Hierarchical Radiosity: A Step towards Frame-to-Frame Coherent Radiosity", Ignacio Martin, Dani Tost, and Xavier Pueyo.

LSI-96-4-R "Skip-Trees, an Alternative Data Structure to Skip-Lists in a Concurrent Approach", Xavier Messeguer.

LSI-96-5-R "Change of Belief in SKL Model Frames (Automatization Based on Analytic Tableaux)", Matías Alvarado and Gustavo Núñez.

LSI-96-6-R "Compressibility of Infinite Binary Sequences", José L. Balcázar, Ricard Gavaldà, and Montserrat Hermo.

LSI-96-7-R "A Proposal for Word Sense Disambiguation using Conceptual Distance", Eneko Agirre and German Rigau.

LSI-96-8-R "Word Sense Disambiguation Using Conceptual Density", Eneko Agirre and German Rigau.

LSI-96-9-R "Towards Learning a Constraint Grammar from Annotated Corpora Using Decision Trees", Lluis Màrquez and Horacio Rodríguez.

LSI-96-10-R "POS Tagging Using Relaxation Labelling", Lluís Padró.

LSI-96-11-R "Hybrid Techniques for Training HMM Part-of-Speech Taggers", Ted Briscoe, Greg Grefenstette, Lluís Padró, and Iskander Serail.

LSI-96-12-R "Using Bidirectional Chart Parsing for Corpus Analysis", A. Ageno and H. Rodríguez.

LSI-96-13-R "Limited Logical Belief Analysis", Antonio Moreno.

LSI-96-14-R "Logic as General Rationality: A Survey", Ton Sales.

LSI-96-15-R "A Syntactic Characterization of Bounded-Rank Decision Trees in Terms of Decision Lists", Nicola Galesi.

LSI-96-16-R "Algebraic Transformation of Unary Partial Algebras I: Double-Pushout Approach", P. Burmeister, F. Rosselló, J. Torrens, and G. Valiente.

LSI-96-17-R "Rewriting in Categories of Spans", Miquel Monserrat, Francesc Rosselló, Joan Torrens, and Gabriel Valiente.

LSI-96-18-R "On the Depth of Randomly Generated Circuits", Tatsuie Tsukiji and Fatos Xhafa.

LSI-96-19-R "Learning Causal Networks from Data", Ramon Sangüesa i Solé.

LSI-96-20-R "Boundary Generation from Voxel-based Volume Representations", R. Joan-Arinyo and J. Solé.

LSI-96-21-R "Exact Learning of Subclasses of CDNF Formulas with Membership Queries", Carlos Domingo.

LSI-96-22-R "Modeling the Thermal Behavior of Biosphere 2 in a Non-Controlled Environment Using Bond Graphs", Angela Nebot, François E. Cellier, and Francisco Mugica.

LSI-96-23-R "Obtaining Synchronization-Free Code with Maximum Parallelism", Ricard Gavaldá, Eduard Ayguadé, and Jordi Torres.

LSI-96-24-R "Memoisation of Categorial Proof Nets: Parallelism in Categorial Processing", Glyn Morrill.

LSI-96-25-R "Decision Trees Have Approximate Fingerprints", Víctor Lavín and Vijay Raghavan.

LSI-96-26-R "Visible Semantics: An Algebraic Semantics for Automatic Verification of Algorithms", Vicent-Ramon Palasí Lallana.

LSI-96-27-R "Massively Parallel and Distributed Dictionaries on AVL and Brother Trees", Joaquim Gabarró and Xavier Messeguer.

LSI-96-28-R "A Maple package for semidefinite programming", Fatos Xhafa and Gonzalo Navarro.

LSI-96-29-R "Bounding the expected length of longest common subsequences and forests", Ricardo A. Baeza-Yates, Ricard Gavaldà, and Gonzalo Navarro.

LSI-96-30-R "Parallel Computation: Models and Complexity Issues", Raymond Greenlaw and H. James Hoover.

LSI-96-31-R "ParaDict, a Data Parallel Library for Dictionaries (Extended Abstract)", Joaquim Gabarró and Jordi Petit i Silvestre.

LSI-96-32-R "Neural Networks as Pattern Recognition Systems", Lourdes Calderón.

LSI-96-33-R "Semàntica externa: una variant interessant de la semàntica de comportament" (written in Catalan), Vicent-Ramon Palasí Lallana.

LSI-96-34-R "Automatic verification of programs: algorithm ALICE", V.R. Palasí Lallana.

LSI-96-35-R "Multiresolution Approximation of Polyhedral Solids", D. Ayala, P. Brunet, R. Joan-Arinyo, I. Navazo.

LSI-96-36-R "Algebraic Transformation of Unary Partial Algebras II: Single-Pushout Approach", P. Burmeister, M. Montserrat, F. Rosselló, and G. Valiente.

LSI-96-37-R "Probabilistic Conditional Independence: A Similarity-Based Measure and its Application to Causal Network Learning", Ramon Sangüesa Solé, Joan Cabós Fabregat, and Ulises Cortés García.

LSI-96-38-R "Analysing the Process of Enforcing Integrity Constraints", Enric Mayol and Ernest Teniente.

LSI-96-39-R "Reducció de l'equivalència inicial visible a teoremes inductius" (written in Catalan), Vicent-Ramon Palasí Lallana.

LSI-96-40-R "A Compendium of Problems Complete for Symmetric Logarithmic Space", Carme Àlvarez and Raymond Greenlaw.

LSI-96-41-R "Semàntica algebraica del llenguatge AL: l'algorisme α" (written in Catalan), V.R. Palasí Lallana.

LSI-96-42-R "Partial Occam's Razor and its Applications", Carlos Domingo, Tatsuie Tsujiki, and Osamu Watanabe.

LSI-96-43-R "Transparent Distributed Problem Resolution in the MAKILA Multi-Agent System", Karmelo Urzelai.

LSI-96-44-R "The Intensional Events Method for Consistent View Updating", Dolors Costal, Ernest Teniente, and Toni Urpí.

LSI-96-45-R "Extending Eiffel as a Full Life-cycle Language", Alonso J. Peralta and Joan Serras.

LSI-96-46-R "Analysis of Methods for Generating Octree Models of Objects from Their Silhouettes", Marta Franquesa and Pere Brunet.

LSI-96-47-R "Learning nearly monotone k-term DNF", Jorge Castro, David Guijarro, and Víctor Lavín.

LSI-96-48-R "Learning Monotone Term Decision Lists", David Guijarro, Víctor Lavín, and Vijay Raghavan.

LSI-96-49-R "Coding Complexity: The Computational Complexity of Succinct Descriptions", José L. Balcázar, Ricard Gavaldà, and Osamu Watanabe.

LSI-96-50-R "Algorithms for Learning Finite Automata from Queries: A Unified View", José L. Balcázar, Josep Díaz, Ricard Gavaldà, and Osamu Watanabe.

LSI-96-51-R "Mètodes de validació d'esquemes de bases de dades deductives" (written in Catalan), Carles Farré.

LSI-96-52-R "The Parallel Complexity of Positive Linear Programming", Luca Trevisan and Fatos Xhafa.

LSI-96-53-R "Polynomial-Time Algorithms for Some Self-Duality Problems", Carlos Domingo.

LSI-96-54-R "Patterns in Random Binary Search Trees", Philippe Flajolet, Xavier Gourdon, and Conrado Martínez.

LSI-96-55-R "El llenguatge ROSES. Part I" (written in Catalan), M. Barceló, P. Costa, D. Costal, A. Olivé, C. Quer, A. Roselló, M.R. Sancho.

LSI-96-56-R "Double-Pushout Hypergraph Rewriting through Free Completions", P. Burmeister, F. Rosselló, G. Valiente.

LSI-96-57-R "On User-Defined Features", Christoph M. Hoffmann and Robert Joan-Arinyo.

LSI-96-58-R "Light Transfer Equations for Volume Visualization", Francesc Sala i Valcàrcel and Daniela Tost i Pardell.

LSI-96-59-R "Computing the Medial Axis Transform of Polygonal Objects by Testing Discs", Josep Vilaplana.

LSI-96-60-R "Linear Lower Bounds and Simulations in Frege Systems with Substitutions", M. Bonet and N. Galesi.

LSI-96-61-R "Prototipado de programas usando especificaciones funcionales y no funcionales" (written in Spanish), Xavier Franch and Pere Botella.

LSI-96-62-R "Transformación de restricciones de integridad dinámicas definidas hacia el futuro en una forma definida hacia el pasado", Maria Amélia Pacheco Silva and Maria Ribera Sancho i Samsó.

LSI-96-63-R "New Results about the List Access Problem", Salvador Roura and Conrado Martínez.

LSI-96-64-R "Orthogonal polyhedra as geometric bounds in constructive solid geometry", A. Aguilera and D. Ayala.

Hardcopies of reports can be ordered from:

Nuria Sánchez
Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Pau Gargallo, 5
08028 Barcelona, Spain
`secrelsi@lsi.upc.es`

See also the Department WWW pages, `http://www-lsi.upc.es/www/`