The Algebra of Shapes

Rudi Stouffs

Dissertation

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

Department of Architecture
Carnegie Mellon University
Pittsburgh, PA 15213

April, 1994
I hereby declare that I am the sole author of this thesis.

I authorize Carnegie Mellon University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Carnegie Mellon University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.
This thesis investigates a new representation scheme for geometric modeling, based on an algebraic model for shapes and formalized using a boundary representation. The algebraic model is mathematically uniform for shapes of all kinds and provides a natural and intuitive framework for mixed-dimensional shapes. The corresponding maximal element representation is essential to the concept of shape emergence. The representation scheme particularly supports computational design as a generative process of search or exploration.

This thesis begins with a treatment of the algebraic and geometric properties of shapes and gives a formal and complete definition of the maximal element representation for $n$-dimensional shapes in a $k$-dimensional space ($n \leq k$). Efficient algorithms are presented for the algebraic operations of sum, product, difference and symmetric difference on shapes of plane and volume segments. An exploration of related research in shape grammars, computational design and construction simulation, illustrates the potential of this representation scheme, while an agenda for future research depicts its present shortcomings.
aan mama en papa,
bedankt
I am indebted to Ramesh Krishnamurti, my advisor and thesis committee chairman, as well as mentor, for his unqualified support throughout the work on this thesis. I also thank my thesis committee, Irving Oppenheim and Michael Erdmann, for their support and contributions.

A number of people have contributed to this thesis, directly or indirectly through their participation on the GRAIL project. Zeyno Aygen is currently developing the GRAIL interface and provided me with some snapshots. Jonah Tsai is also a current member in the GRAIL team. Marc Schindewolf, Robert Tseng and Churn-Der Hwang (as well as myself) pioneered the GRAIL project under the supervision of Ramesh Krishnamurti.

I also wish to thank all my friends for making my stay in Pittsburgh so enjoyable.

Most of all, I thank my family for making this adventure possible. My parents, Lode and Thérèse, especially deserve my love and gratitude.
# Table of Contents

## List of Figures

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## List of Tables

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Chapter 1 Introduction

1.1 An Algebra and Geometry for Shapes, 3  
1.2 Research Background, 4  
1.2.1 Shape Grammars, 5  
1.2.2 Geometric Modeling, 6  
1.3 Demonstration, 9  
1.4 Overview, 9

## I Model and Representation

## Chapter 2 Algebras of Shapes

2.1 Algebraic Model, 16  
2.1.1 Definitions, 16  
2.1.2 Generalized Boolean Algebra, 21  
2.1.3 Boolean Ring, 28  
2.2 Shape Relations, 37
# Table of Contents

## Chapter 3  Geometry of Shapes  ................................................. 39

3.1 Shape Representation, 40  
3.1.1 Co-descriptor, 40  
3.1.2 Boundary, 42  
3.1.3 Segment, 50  
3.2 Maximal Segment Representation, 53  
3.2.1 Canonical Representation, 54  
3.2.2 Geometric Relations, 62  
3.2.3 Maximal Shapes, 63

## Chapter 4  Geometric Modeling .................................................. 67

4.1 Shapes, 68  
4.1.1 Representation, 69  
4.1.2 Cross-Algebra Operations, 75  
4.2 Solid Modeling, 76  
4.2.1 Solids, 76  
4.2.2 Isomorphisms, 79  
4.2.3 Comparison, 87

### II Arithmetic and Algorithms

## Chapter 5  Boundary Evaluation .................................................. 93

5.1 Boundary Evaluation and Merging, 94  
5.2 Shape Arithmetic, 95  
5.2.1 Boundary Classification, 95  
5.2.2 Shape Operations, 101  
5.2.3 Shape Relations, 106

## Chapter 6  Classification Algorithms .......................................... 113

6.1 Preliminaries, 114  
6.2 Classification of Line Segments, 117  
6.3 Classification of Plane Segments, 131

## Chapter 7  Construction Algorithms ............................................ 141

7.1 Construction of Plane Segments, 142  
7.2 Construction of Volume Segments, 151

## Chapter 8  Complexity Analysis .................................................. 173

8.1 Computational Complexity, 174  
8.1.1 Time Complexities, 175  
8.1.2 Comparison, 181  
8.2 Robustness, 184
## III Past, Present and Future

### Chapter 9 Shape Emergence

- 9.1 Shape Grammars, 192
  - 9.1.1 Reversible and Irreversible Rules, 194
- 9.2 Subshape Detection, 199
  - 9.2.1 Shape Recognition in $U_{n,3}$, 200
  - 9.2.2 Shape Recognition Revisited, 205
  - 9.2.3 Variations on a Common Theme, 216

### Chapter 10 Applications

- 10.1 RUBICON, 219
  - 10.1.1 The RUBICON Project, 220
  - 10.1.2 Collision Avoidance, 226
- 10.2 GRAIL, 228
  - 10.2.1 Demonstration, 230

### Chapter 11 Conclusion

- 11.1 Contributions, 238
- 11.2 An Agenda for Future Research, 238
  - 11.2.1 Surfaces and Curves, 239
  - 11.2.2 Augmented Shapes, 240

### Glossary of Mathematical Notation

### Bibliography
Table of Contents

X
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>A shape consisting of 6 maximal line segments.</td>
<td>3</td>
</tr>
<tr>
<td>1.2</td>
<td>Rule and rule application.</td>
<td>4</td>
</tr>
<tr>
<td>1.3</td>
<td>The intersection of two three-dimensional point sets may result in a two-</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>dimensional point set.</td>
<td></td>
</tr>
<tr>
<td>1.4</td>
<td>A computation using the operations of sum, product and difference, the part</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>and equality relations, and the translation operator.</td>
<td></td>
</tr>
<tr>
<td>2.1</td>
<td>Two shapes $a$ and $b$ and their product $a \cdot b$.</td>
<td>18</td>
</tr>
<tr>
<td>2.2</td>
<td>Two shapes $a$ and $b$ and their differences $a - b$ and $b - a$.</td>
<td>19</td>
</tr>
<tr>
<td>2.3</td>
<td>Two shapes $a$ and $b$ and their sum $a + b$.</td>
<td>20</td>
</tr>
<tr>
<td>2.4</td>
<td>Three shapes $a$, $b$ and $c$ and their sum $a + b + c$.</td>
<td>22</td>
</tr>
<tr>
<td>2.5</td>
<td>Three shapes $a$, $b$ and $c$ and their product $a \cdot b \cdot c$.</td>
<td>23</td>
</tr>
<tr>
<td>2.6</td>
<td>Three shapes $a$, $b$ and $c$; the shapes $b + c$, $a \cdot b$ and $a \cdot c$; and the shape $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$.</td>
<td>27</td>
</tr>
</tbody>
</table>
Figure 2.7  Two shapes $a$ and $b$ with $a \leq b$, the relative complement, $b - a$, of $a$ with respect to $b$ and the shapes $a \cdot (b - a)$ and $a + (b - a)$. ........................................ 28

Figure 2.8  Two shapes $a$ and $b$ and their “disjoint” parts $a - b$, $b - a$ and $a \cdot b$. ... 30

Figure 2.9  Two shapes $a$ and $b$ and their symmetric difference $a \oplus b$. .............. 30

Figure 2.10  Two shapes $a$ and $b$, their sum, product, differences and symmetric difference. ......................................................... 32

Figure 2.11  Graphical description of the partially ordered set of shapes derived from the shapes $a$ and $b$ using the operations of sum, product, difference and symmetric difference. ........................................ 33

Figure 2.12  Three shapes $a$, $b$ and $c$ and their symmetric difference $a \oplus b \oplus c$. .... 35

Figure 2.13  The shape relations contain, overlap and disjoint. ......................... 38

Figure 3.1  Two shapes $a$ and $b$ and a part $s$ of the boundary of $a$, resulting in three possible cases upon partitioning $s$ and $a$ with respect to $b$. .................... 47

Figure 3.2  Two shapes $a$ and $b$, the sum of their boundaries $B[a] + B[b]$ and the boundaries of their sum, product, differences and symmetric difference. ................. 49

Figure 3.3  Exemplar segments in $U_2$. ......................................................... 51

Figure 3.4  Line segments. ................................................................. 51

Figure 3.5  A shape $a$, a representation $\{a_1, ..., a_m\}$ of $a$ and the symmetric difference of the boundaries $B[a_1] \oplus ... \oplus B[a_m]$. ................................. 52

Figure 3.6  The sum of two segments is a segment if the segments are not disjoint or their boundaries are not disjoint and is not a segment, otherwise. .......... 54

Figure 3.7  Examples of segments with and without a simple boundary. .......... 57

Figure 3.8  Inner and outer boundaries among a set of non-intersecting, simple boundaries (in $U_1$). ................................................................. 59

Figure 3.9  Outer and inner boundaries for a few shapes (in $U_2$). ............... 59

Figure 3.10  Two different interpretations for the boundary of a shape in $U_2$. ..... 61

Figure 3.11  The geometric shape relations contain, overlap, share boundary and disjoint. ................................................................................. 63

Figure 3.12  Two special cases. ................................................................. 64
List of Figures xiii

Figure 4.1 A line segment $l$ with endpoints $p$ and $q$, the direction vector and moment vector about the origin $0$ of the carrier of $l$. ................................. 70
Figure 4.2 A plane segment $f$ with boundary line segments $l$ and $l'$, the normal vector and normal distance of the carrier of $f$. ................................. 72
Figure 4.3 The orientation of the inside of a plane segment $f$ with respect to a boundary line segment $l$. ............................................................ 74
Figure 4.4 Two co-equal boundary line segments $l$ and $l'$ of a plane segment $f$, that share boundary and have respective opposite orientations of the inside of $f$ . . . 75
Figure 4.5 Regularity of solids. ................................................................. 77
Figure 4.6 The difference and intersection of two regular point sets in $E^3$ that share a 2-dimensional subset. .............................................................. 78
Figure 4.7 Manifold versus nonmanifold. ................................................... 78
Figure 4.8 A line segment $l$ is an element of $L_s$ if the normal vector indicating the inside of $s$ with respect to $f_l$ and the reference vector make an acute angle. . . . 83
Figure 4.9 An endpoint $p$ belongs to the set $P_s$ if either $f_p \in F_s$, in the case of a single plane segment $f_p$ corresponding to $k_p$ or $k_p \in L_s$, otherwise. ......................... 84
Figure 4.10 Two solids $a$ and $b$ that have overlapping faces with equal “insides”; their union, intersection and difference. .................................................. 85
Figure 4.11 Two solids $a$ and $b$ that have overlapping faces with opposite “insides”; their union, intersection and difference. .................................................. 86
Figure 4.12 Examples to illustrate the validity of the constructed isomorphism for the edges of two solids under the Boolean set operation of sum. ................................. 86
Figure 4.13 A shape in $U_1 \times U_2 \times U_3$ is made up of subshapes in $U_1$, $U_2$ and $U_3$ that may “overlap” or “intersect”, but that do not compose. ................................. 88

Figure 5.1 Two shapes $a$ and $b$ and their shared boundary segments. .......... 96
Figure 5.2 The same-shared and opposite-shared boundary segments of two shapes $a$ and $b$. ............................................................... 97
Figure 5.3 The boundary segments of either of two shapes $a$ and $b$ that are deemed inner with respect to the other shape. .......................................... 99
Figure 5.4 The boundary segments of either of two shapes $a$ and $b$ that are deemed outer with respect to the other shape. .......................................... 99
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.5</td>
<td>Inner and outer neighborhoods of a boundary segment ( l ) of ( a ) with respect to shapes ( a ) and ( b ).</td>
<td>103</td>
</tr>
<tr>
<td>5.6</td>
<td>Inner and outer neighborhoods of a boundary segment ( l ) of ( a ) with respect to shapes ( a ) and ( b ).</td>
<td>104</td>
</tr>
<tr>
<td>5.7</td>
<td>Examples showing that each of the conjunctions i through vii is necessary as a (sufficient) condition for two shapes to overlap.</td>
<td>109</td>
</tr>
<tr>
<td>6.1</td>
<td>Simple cases of intersection of an infinite line with the boundary of a shape in ( U_2 ).</td>
<td>118</td>
</tr>
<tr>
<td>6.2</td>
<td>All degenerate cases are resolved by translating the line of intersection over an infinitesimal distance perpendicular to its axis.</td>
<td>119</td>
</tr>
<tr>
<td>6.3</td>
<td>Four cases for the intersection of a line segment with a line.</td>
<td>120</td>
</tr>
<tr>
<td>6.4</td>
<td>Given a line segment ( k ), with endpoints ( p ) and ( q ), and a line ( l ) through ( p ), we can determine on which side of ( l ) a point ( q ) lies by computing the equation of a plane normal to ( f ) and through ( l ), where ( f ) is the plane defined by ( l ) and the carrier of ( k ).</td>
<td>120</td>
</tr>
<tr>
<td>6.5</td>
<td>The line of intersection of two planes is uniquely determined by its direction vector and any point on the line.</td>
<td>122</td>
</tr>
<tr>
<td>6.6</td>
<td>The pairs of line segments to be examined for forthcoming intersection points, for each of the four basic types of transition points.</td>
<td>127</td>
</tr>
<tr>
<td>6.7</td>
<td>Comparison of the location of a point ( p ) with respect to a line segment ( l ).</td>
<td>128</td>
</tr>
<tr>
<td>6.8</td>
<td>Four cases for the intersection of a plane segment with a plane.</td>
<td>132</td>
</tr>
<tr>
<td>6.9</td>
<td>Given a plane segment ( g ) and a plane ( f ) through a boundary segment ( l ) of ( g ), we can determine the “side” of ( g ) with respect to ( f ) by computing the vector perpendicular to the direction vector of ( l ) and the normal vector of ( g ).</td>
<td>133</td>
</tr>
<tr>
<td>6.10</td>
<td>A shape in ( U_3 ) consisting of ( 4^3/2 = 32 ) segments (cubes), ( 4^3<em>3 = 192 ) boundary plane segments, but only ( 3</em>(4+1) = 15 ) co-equal classes of boundary segments.</td>
<td>137</td>
</tr>
<tr>
<td>7.1</td>
<td>Two different interpretations for the boundary of a plane segment.</td>
<td>143</td>
</tr>
<tr>
<td>7.2</td>
<td>The role of coinciding line segments in the determination of simple boundaries.</td>
<td>143</td>
</tr>
</tbody>
</table>
List of Figures

Figure 7.3 Clockwise and counterclockwise configurations of three line segments $a$, $b$ and $c$ about a common endpoint, when all angles are less than $180^\circ$ or a single angle is greater or equal to $180^\circ$. ........................................ 144

Figure 7.4 Two examples illustrating the start and continuation steps in the outlined algorithm to extract the simple polygons from a set of boundary segments. .................................................... 145

Figure 7.5 An exemplar result of the procedure CYCLES, consisting of four simple cycles, of which all but one represent an inner boundary for the defined shape. 146

Figure 7.6 The configuration of three plane segments $f$, $g$ and $h$ about a common boundary line segment $l$. .......................................................... 152

Figure 7.7 Illustrations of a left-most line segment $l$ and the bottom direction plane $b$ with respect to $l$. .......................................................... 153

Figure 7.8 Partial shells. .......................................................... 155

Figure 7.9 Shells. .......................................................... 156

Figure 7.10 A shape defined by one outer boundary and three inner boundaries. . . . 161

Figure 7.11 The enclosure-tree of an exemplar shape defined by six simple boundaries. .......................................................... 163

Figure 7.12 A point on the inside of a boundary segment $k$ of a plane segment $g$ is a valid piercing point if … .................................................... 164

Figure 7.13 An endpoint of a boundary segment $k_j$ of a plane segment $g$, where $k_j$ makes the smallest angle with the line of intersection $m$ of $g$ and a reference half-plane $f$, is a valid piercing point if … .................................................... 165

Figure 8.1 (Ir)rational transformations. ........................................ 186

Figure 8.2 A composition of a $30^\circ$ rotation with an arbitrary scaling is always irrational, even in two dimensions. ........................................ 187

Figure 9.1 Exemplar derivations from an initial shape with a single rule. . . . . . 195

Figure 9.2 An example of an irreversible rule $a \rightarrow b$. ........................................ 197

Figure 9.3 An example of a reversible rule $a \rightarrow b$. ........................................ 197

Figure 9.4 Examples illustrating the determinate cases for $U_{1,3}$. ............... 201
Figure 9.5  Examples illustrating the indeterminate cases for $U_{1,3}$. ............... 202
Figure 9.6  Examples illustrating the determinate Case 2.1 for $U_{2,3}$. ............... 203
Figure 9.7  Examples illustrating the indeterminate cases for $U_{2,3}$. ............... 204
Figure 9.8  The base transformation maps carriers onto carriers without scaling. . . 205
Figure 9.9  Illustrations to Case I.b. ....................................................... 210
Figure 9.10 Illustrations to (a) Case I.c and (b) Case I.d. ................................. 211
Figure 9.11 Illustrations to (a) Case I.i and (b) Case I.j. ................................. 212
Figure 9.12 Illustrations to Case I.j. ....................................................... 213
Figure 10.1  A robot crane. ................................................................. 221
Figure 10.2  A robot towmotor with stationary rotational motion and translational motion along its axis. ................................................................. 221
Figure 10.3  Front and side view of a possible configuration of an exterior wall finishing robot. ................................................................. 222
Figure 10.4  Three frames to describe the configuration of a robot. ................. 222
Figure 10.5  Typical 40 unit building during construction. ............................. 225
Figure 10.6  Snapshot of the RUBICON graphical user interface. ................. 226
Figure 10.7  Dual robot task plan (snapshots from the simulation). ................. 227
Figure 10.8  Creation tools for line and plane segments. ............................... 231
Figure 10.9  Different drawing modes allow for orthogonal snapping, among others. 231
Figure 10.10 Operation modes when merging the selection shape with the design, e.g., “difference”. ............................................................... 232
Figure 10.11 Different modes specify the status of the selection shape after merging (“sum”) it with the design. ............................................................... 232
Figure 10.12 “Cutter” selection tools. ........................................................... 233
Figure 10.13 The “switch” tool switches the selection and design shapes. ........... 233
Figure 10.14 The “bin” tool discards the current selection shape. ..................... 234
Figure 10.15 The transformation tools are translation, reflection, scaling and rotation. 234
Figure 10.16  The “copy” mode is currently activated: the result of the translation is a copy of the selection shape. .................................................. 235

Figure 10.17  Reflection about a line (without copy). ................................. 235

Figure 10.18  Rotation. ................................................................. 236

Figure 11.1  The operations of sum and product over weighted shapes of line segments. ................................................................. 241
List of Figures xvi

dAcknowledgements xix

dIntroduction xi
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 3.1</td>
<td>Mappings and representation for each of the algebras $U_n$.</td>
<td>40</td>
</tr>
<tr>
<td>Table 5.1</td>
<td>Set-membership classification for the Boolean set operations of union, intersection and difference into the 8-way boundary classification sets.</td>
<td>95</td>
</tr>
<tr>
<td>Table 5.2</td>
<td>Set-membership classification for the operations of sum, product, difference and symmetric difference, applied to two shapes $a$ and $b$.</td>
<td>102</td>
</tr>
<tr>
<td>Table 5.3</td>
<td>Set-membership conditions for the relations contain, share boundary and disjoint, applied to two plane segments $a$ and $b$.</td>
<td>107</td>
</tr>
<tr>
<td>Table 5.4</td>
<td>Set-membership condition sets for the relation $overlap$, applied to two plane segments $a$ and $b$.</td>
<td>107</td>
</tr>
<tr>
<td>Table 6.1</td>
<td>The algebras $U_{n,k}$ under consideration.</td>
<td>114</td>
</tr>
<tr>
<td>Table 7.1</td>
<td>Combinations of clockwise and counterclockwise angles for each of the angles $\angle ab$, $\angle bc$ and $\angle ca$, grouped with respect to the overall ordering $(a, b, c)$ or $(a, c, b)$.</td>
<td>144</td>
</tr>
</tbody>
</table>
Table 8.1  The time complexities of the shape operations in the algebras $U_k$, $0 \leq k \leq 3$. ................................................................. 180

Table 9.1  Ten possible combinations of independent transformations corresponding to one determinate and nine indeterminate cases for the subshape recognition problem in $U_0 \times U_1 \times U_2 \times U_3$. ................................................. 207

Table 9.2  All non-redundant cases with at least one distinguishable point, based on combinations of primary distinguishable elements. .................. 208

Table 9.3  All non-redundant cases without distinguishable points but at least one distinguishable line, based on combinations of primary distinguishable elements. ......................................................... 209

Table 9.4  All non-redundant cases with only distinguishable planes. .............. 210
I investigate a new representation scheme for geometric modeling based on a purely algebraic model and formalized using a boundary representation. In this thesis, I focus on a modeling space consisting of $n$-dimensional rectilinear shapes of finite but non-zero measure. The model is also applicable to curved shapes, and is extensible to non-geometric elements and attributes. The global operations of sum, product, difference and symmetric difference on shapes, similar though not identical to the Boolean set operations of union, intersection, difference and symmetric difference on point sets, define an algebra of shapes.

The algebraic model is based on a part relation so that any part of a shape is also considered a shape. The part relation is formalized in the maximal element representation. This representation is particularly suited to answer the following two questions: Are two shapes identical? and Is one shape a subshape or part of another shape? The representation scheme is both unique and unambiguous.

The algebraic model and the corresponding maximal element representation for shapes have their origin in shape grammars research (Stiny, 1980a, 1986, 1991). However, grammars
Introduction

constitute only a particular formalized subset of their potential applications. The model has high potential for applicability to design in general and to architectural design in particular (even in the early design stages) and is especially suited to support design search or exploration.

In this thesis, I draw upon and extend current shape grammars research, present the algebraic model and the maximal element representation as a representation scheme for geometric modeling and relate this research to computational design. The main contributions are:

- A representation scheme for shapes. I formally prove the algebraic and geometric properties of shapes. I give a formal and complete definition of the maximal element representation for \( n \)-dimensional shapes in a \( k \)-dimensional space (\( n \leq k \)). I show that an isomorphism exists between shapes and point sets in the Euclidean space.

- Efficient algorithms for shape arithmetic. I extend the maximal element representation to shapes of volume segments, under the algebraic operations of sum, product, difference and symmetric difference. I relate the algebraic operations on \( n \)-dimensional shapes to \((n-1)\)-dimensional arithmetic on their boundary shapes. I develop efficient algorithms for the arithmetic on 2- and 3-dimensional rectilinear shapes, based on a classification of boundary segments. I show that the computational complexity of these algorithms is comparable to complexity analysis results presented in computational geometry literature.

- An exploration of related research. In relation to the shape grammars formalism, I exhaustively enumerate the possible cases for shape recognition on mixed-dimensional shapes. I present two exemplar applications of the representation scheme for shapes to illustrate its potential impact on computational design.

- An agenda for future research necessary to give this representation scheme the power to compete with other geometric models, in particular, an extension to surfaces and curved elements and the inclusion of non-geometric attributes.
1.1 An Algebra and Geometry for Shapes

The algebraic model is based on a part relationship. This part relation can be freely defined as long as it constitutes a partial order relation. For practical purposes, it is advantageous to define the part relation between elements of the same type, e.g., between elements of the same dimensionality. Fundamental to the algebraic model is that, under the part relation, any part of a shape is a shape. As such, a shape defines an infinite set of (sub)shapes that are each part of the original shape. The maximal element representation for shapes captures this notion (Krishnamurti, 1992a).

Figure 1.1 illustrates this definition of a shape, under the part relation, with a shape of maximal line segments\(^1\). The six line segments can be grouped three by three to form two triangles. Other interpretations are possible; up to five triangles can be recognized in the shape and manipulated as such. Figure 1.2 shows a rule that rotates an equilateral triangle a half-turn about its center and repeated applications of the rule to a shape composed of three triangles. The resulting shape is a rotation of the initial shape about its center (Stiny, 1993). This simple computation illustrates how a single shape can be interpreted in a variety of different ways. Shapes that were not originally envisioned, emerge under the part relation. A rule, then, defines a shape transformation on (emergent) subshapes detected under the part relation. The concept of emergent shapes is highly enticing to design search; the specification of such shape transformations leads naturally to design generation and search.

---

1. A segment is maximal if it cannot be combined with any other segment in the same shape to form a single segment, under the operation of sum or symmetric difference (Definition 3.12).
We consider a shape an element of an algebra that is ordered by a part relation and closed under the operations of sum, product and difference and the affine transformations. Elements of the same dimensionality belong to the same algebra. A shape may consist of more than one type of element, in which case it belongs to the algebra given by the Cartesian product of the algebras of its element types. The parallel treatment of spatial elements with different dimensionality allows for the representation - within a single space - of compound shapes as complex objects made up of segments from different algebras, or as consisting of shapes that are coordinated and related (see also Section 1.2.2).

1.2 Research Background

The work described in this thesis lies within the fields of computational design and geometric modeling. Computational design is in many respects still in its infancy. Commercially, computer-aided design is (often) nothing more than a euphemism for

---

2. There is not a single algebra of shapes. The title of this thesis, “The Algebra of Shapes,” refers to the algebraic description of shapes. This description includes the geometry of shapes.
Introduction

computer-aided drafting, and a CAD system serves merely as an electronic repository for the transference of designed information. Fundamental changes are required if we want to support the design process in all its aspects, especially in the early design stages. The paradigms of design as search (Akin, 1986; Woodbury, 1991) and design as exploration (Smithers and Troxell, 1990) offer exciting promise in this respect. Geometric and solid modeling, on the other hand, are well established and, though far from exhausted, can rely on a solid basis of proven techniques and previous research. Two different model types are commonly used: constructive models make use of global operations to build solids from primitives; boundary models use local operations to construct the boundary of a solid (Mäntylä, 1988; Hoffmann, 1989a). From this perspective, it may seem a little odd (and perhaps, ironical) that in this thesis I attempt to instill the field of geometric modeling with a fundamentally new approach that draws its origins from recent computational design research. Through rigorous analysis and construction of this new representation scheme, I will show its potential for geometric modeling and its advantages over other schemes.

1.2.1 Shape Grammars

Design as generation fits particularly well in a computational formalism for design search. A generation scheme usually consists of a set of generation rules that can be applied to an initial object in order to generate a variety of solutions or design objects. The set of rules, together with the initial object and the vocabulary of elements from which the rules are composed, is termed a grammar and the objects resulting from a generative application of such a grammar constitutes the derived language. Specifically, a shape grammar is a formal rewriting system for producing languages of shapes (Stiny, 1980a; see also Chapter 9).

Grammars can be used for both analysis and generation (or synthesis). So far, examples in analysis have been more prominent (Stiny and Mitchell, 1978; Knight, 1980; Koning and Eizenberg, 1981; Flemming, 1987). Generation can assist the designer in the exploration of design alternatives by taking over part of the generation and transformation process along one or more paths as outlined by the designer. If powerful enough in its representation, it can also assist the designer in converting a concept or idea into a more comprehensible or tangible design. In the extreme, it may be useful simply to expedite some repetitive drafting
tasks. Far from being an exhaustive list, these examples point to the range of abilities a generational tool can bring to a computational design environment.

At the same time, a generational tool may also impose some restrictions, if not on the other parts of the computational design environment, at least on the usage of this tool in the design process. As powerful as the concept of a grammar and its corresponding language of design may be, it imposes some restrictions on the usage of the generational approach: A single rule becomes insignificant as far as it does not form part of a grammar and the concept of generation or, more generally, search exists only within the confinements of a grammar.

However, the concept of search is more fundamental to design than its generational form alone might imply. Any mutation of an object into another one, or parts thereof, whether as the result of a transformation or operation, constitutes an action of search. A rule constitutes a particular compound operation or mutation, that is, a composition of a number of operations and/or transformations that is recognized as a new, single, operation and can be applied as such. Similarly, a grammar is just a collection of rules or operations that yields a certain set (or language) of design objects given an initial object. As such, the creation of a grammar is only a tool that allows a structuring of a set of operations that has proven its applicability to the creation of a certain set of objects, rather than a framework for generation.

Particular to a rule is that the transformation is not specified but is selected from a body of transformations, e.g., the similarity transformations, according to a (specified) constraint. For a shape rule, the constraint specifies that the object of mutation is a part of the given shape. The process of determining one (or all) transformation for which this constraint holds is termed subshape detection (see also Chapter 9). It relies fundamentally on the algebraic model for shapes and the corresponding maximal element representation.

### 1.2.2 Geometric Modeling

Solid or geometric modeling (Mäntylä, 1988; Hoffmann, 1989a) is generally committed to an Euclidean space. Even though the Euclidean space is advantageous to a representation, it introduces certain anomalies with respect to solids, such as the notion of regularity and the
regular variants of the Boolean set operations (Figure 1.3; see also Chapter 4). Under the algebraic model, the operations of sum, product and difference (corresponding to the Boolean set operations of union, intersection and difference on point sets) are defined using the part relation which is itself defined only between spatial elements of the same dimensionality. As such, the algebraic operations are intrinsically regular.

The absence of a part relation between elements with different dimensionality is more important when dealing with shapes of mixed dimensionality. In the field of mechanical design, a solution usually consists of a composition of standardized elements, that has been designed in the same vein of thought. Architectural design on the other hand reflects more than the resulting composition of bricks or timber. In general, even though the final built product is a composition of (purely) three-dimensional solid elements, the creative process of design is not as much concerned with the dimensionalities of each and every individual element. Even in the evaluation of a design, an abstraction is often more valuable. Structurally, walls can be represented as plane segments with simple integral attributes; a true three-dimensional model would, unless approximated, unnecessarily complicate a structural evaluation. In general, even a single component may be represented as different elements of mixed dimensionality, each element projecting information for a specific application.

Mixed-dimensional models have found recent support in solid modeling. Rossignac and Requicha (1991) offer a nonregular approach that allows for the representation of dangling

\[ A \cap B \]

Figure 1.3 The intersection of two three-dimensional point sets may result in a two-dimensional point set.

3. A solid is regular if closed and there exist no “dangling” or “isolated” faces, edges or points.
faces and edges by extending the definition of a valid point set. At the expense of intuition in operations and ease of conception, it allows for almost arbitrary point sets. Gursoz et al. (1991) allow for objects of different dimensionalities, each regular in its own dimensionality. This results in operations that are defined across dimensionalities, even if these may not seem intuitive. For example, the notion of solids touching is completely absent in this representation, as illustrated in Figure 1.3.

Independent of any particular representation, the Euclidean space implies some notion of regularity, influencing the modeling aspect of the design process. To deprive “regularity” of its authority or omnipotence requires a leap of abstractness as characterizes the field of algebra: such conceptually simple and familiar operations as addition and subtraction are applied to an extraordinary variety of elements that often seem rather unrelated. In the odd case when such operations may actually be inconceivable or highly inappropriate, algebra presents us with a quintessential extension, namely, the Cartesian product of algebras. By virtue of the Cartesian product, algebras can be used and applied in combination without mutual interference. This parallel construction allows for objects of quite disparity not only to coexist peacefully in a single (spatial modeling) world but also to be conceived as one at the same time as being handled and operated on quite differently, yet using a conceptually unified approach.

Under the algebraic model, the Euclidean space only serves to embed the elements in their final representation. Regularity and therefore nonregularity, are completely absent in the algebraic model: when embedding elements of different types in the same Euclidean space, these elements cohabit without interference, even though their corresponding point sets may intersect. Formally, a shape consisting of spatial elements of different types constitutes an element of the algebra given by the Cartesian product of the algebras of its spatial element types. Such a shape is compositely embedded in a Cartesian product of Euclidean spaces. These may ultimately be visualized as either a single space or a multiple of spaces. The Cartesian product of algebras is not restricted to a composition of different algebras. Generalized as such, compound shapes can be regarded either as complex shapes that are composed of segments from different algebras or as consisting of shapes that are coordinated or related. For example, consider a set of drawings from amongst plans, elevations and sections of a same building. Each drawing can be considered a shape, as an
element of a two-dimensional algebra, while the set of drawings can be regarded as a single
shape that is an element of a Cartesian product of algebras (here, all of the same
dimensionality).

1.3 Demonstration

Figure 1.4 illustrates the application of the operations of sum, product and difference, the
part and equality relations, and the translation operator, in the computation of a few three-
dimensional shapes.

1.4 Overview

A representation scheme is a relation between a modeling space and a representation space.
We consider a mathematical modeling space composed of shapes of limited but non-zero
measure. We distinguish two aspects in the definition of a shape, one algebraic and the other
geometric: a shape is an element of an algebra that is ordered by a part relation and closed
under the operations of sum, product and difference; a shape is represented as a set of
maximal segments, with a carrier mapping to the Euclidean space and a boundary mapping
to one lower-dimensional algebra.

This thesis is divided into three parts.

Part I defines the representation scheme and compares it with existing solid modeling
schemes. The algebra of shapes is described in Chapter 2. It is proven that the least set of
all shapes with the same dimensionality defines a generalized Boolean algebra or,
equivalently, a Boolean ring. Chapter 3 specifies the geometry of shapes: the model is
completed with a definition of a shape in terms of its composing (maximal) elements and
their boundaries. Chapter 4 concludes the description of the representation scheme with an
exact specification of the representation of a shape and a comparison of the representation
scheme with common solid modeling schemes. It is shown that an isomorphism exists
between shapes and (regular) point sets in the Euclidean space.
Figure 1.4 A computation using the operations of sum “+”, product “⋅” and difference “−”, the part “≤” and equality “=”, relations, and the translation “T()” operator.
The theory exposed in part I is put to the test in part II in an exposition of the algorithmic aspects of shape arithmetic. We adopt the classification approach of boundary segments as a unified approach to each of the shape operations and relations. Chapter 5 formalizes the approach and outlines the necessary algorithms. Chapters 6 and 7 give a detailed description of the algorithms involved. The classification algorithms are contained in Chapter 6. Upon classification, the resulting shape is constructed from the boundary segments contained in the appropriate classes. The corresponding construction algorithms are given in Chapter 7. In both chapters, special attention is given to the computational complexities of the algorithms. Chapter 8 compares these complexity results with algorithms specified in the literature of computational geometry and solid modeling.

Part III looks at the applicability of the representation scheme in the areas of shape grammars, computational design and construction robotics. Chapter 9 presents new results with respect to shape rule application and subshape detection, in particular. Chapter 10 describes two computational applications of the representation scheme. RUBICON, a rule based simulation applied to robotized building construction, illustrates the applicability of this research to the field of construction robotics and simulation. GRAIL presents an exploration of the challenges and the implications that the algebraic model of shapes poses to computational design. Chapter 11 concludes this thesis with an agenda for future research necessary to give this representation scheme the power to compete with established schemes in geometric modeling.
Part I

Model and Representation
Chapter 2
Algebras of Shapes

A shape is a finite arrangement of (rectilinear) spatial elements from among points, line, plane or volume segments, or higher dimensional hyperplane segments, of limited but non-zero measure. A shape may be considered an element of an algebra \( U \) that is ordered by a part relation and closed under the operations of sum, product and difference\(^1\), and the affine transformations. If the shapes are defined in a \( k \)-dimensional space, \( k \geq n \), \( U_{n,k} \) denotes the set of all finite arrangements of \( n \)-dimensional hyperplane segments of limited but non-zero measure in a \( k \)-dimensional space. If \( k \) is unambiguously understood, it may be dropped and \( U_{n,k} \) can be referred to as \( U_n \). Thus, \( U_0 \) refers to an algebra of shapes made up of points, \( U_1 \) an algebra of shapes made up of line segments, \( U_2 \) an algebra of shapes made up of plane segments and \( U_3 \) an algebra of shapes made up of volume segments, and so on. A shape may consist of more than one type of spatial element, in which case it belongs to the algebra given by the Cartesian product of the algebras of its spatial element types. (Stiny, 1991)

---

\(^1\) Precise definitions for the part relation and the operations of sum, product and difference are given in the sequel.
In this chapter, we develop an algebraic model for shapes in $U_n$, based on the part relation, $\leq$. A shape is a part of another shape if it is embedded in the other shape as a smaller or equal element. This relation is a partial order relation and, as such, $(U_n, \leq)$ is a partially ordered set. Any finite subset of $U_n$ has a greatest lower bound as well as a least upper bound. Given a set of two shapes, the product of the two shapes constitutes the greatest lower bound of this set, while the sum of both shapes constitutes the least upper bound. The empty shape, 0, constitutes the lower bound for $U_n$. $U_n$ does not have an upper bound. Under the operations of sum and product, $(U_n, \leq)$ is a relatively complemented distributive lattice and $(U_n, (+, \cdot, 0))$ defines a generalized Boolean algebra. The relative complement of one shape with respect to another shape is the result of the operation of difference. We define the symmetric difference of two shapes $a$ and $b$ as $a \oplus b = b \oplus a = (a - b) + (b - a)$. Then, $(U_n, (\oplus, \cdot, 0))$ forms a Boolean ring.

Also, we define the following relations among shapes in $U_n$: A shape $a$ is said to contain a shape $b$ if $b$ is a part of $a$. Two shapes overlap if their product is non-zero and neither shape contains the other. Otherwise, the two shapes are considered to be disjoint.

## 2.1 Algebraic Model

The algebraic model we adopt for defining shapes is based on the part relation. The operations of sum, product and difference are defined using the part relation. In the following, all shapes are assumed to be elements of $U_n$. We refer to Balbes and Dwinger (1974) for a more general exposition on algebraic lattices and their properties.

### 2.1.1 Definitions

We define the part relation and show it is a partial order relation. The partial order on $U_n$ is crucial in establishing the algebra of $U_n$.

**Definition 2.1** A *part* is an embedding of one shape in another shape.
The definition of embedding is algebra-specific and depends on the specific representation adopted. A part of a shape is also called a subshape, and specifies the relation ≤. The part relation is a partial order relation. It has the following properties (stated without proof).

\[ \forall a, b, c \in U_n : \]

- Reflexive: \[ a \leq a \] [eq2.1]
- Anti-symmetric: \[ a \leq b \land b \leq a \iff a = b \] [eq2.2]
- Transitive: \[ a \leq b \land b \leq c \iff a \leq c \] [eq2.3]

\((U_n, \leq)\) is a partially ordered set. When an element \(a\) is a part of another element \(b\), it is said that \(b\) contains \(a\). From the transitivity of the part relation, the following property holds.

**Property** A shape is a part of another shape if and only if any part of that shape is a part of the other shape;

\[ \forall a, b: a \leq b \iff (\forall \varepsilon : \varepsilon \leq a \implies \varepsilon \leq b) \] [eq2.4]

*Proof:* Given \(a \leq b\), it follows from [eq2.3] that for all \(\varepsilon \leq a\), \(\varepsilon \leq b\) holds. Conversely, from \(a \leq a\) it follows that \(a \leq b\). \(\square\)

\(U_n\) has a minimal element, the zero element of \(U_n\), denoted 0, the empty shape. The set \(U_n\) of shapes is bounded by an extremum below.

**Definition 2.2** The empty shape 0 is a part of every shape; \(0 \leq \varepsilon, \forall \varepsilon \in U_n\).

**Property** The zero element is the minimal element of \(U_n\), that is, no other element of \(U_n\) is a part of 0;

\[ \varepsilon \leq 0 \iff \varepsilon = 0 \] [eq2.5]

*Proof:* Follows directly from the above definition and [eq2.2]. \(\square\)

---

2. The representation of shapes is discussed in Chapter 3. The embedding of points is defined in Section 3.1.1, the embedding of line segments in Section 3.1.3 and the embedding of shapes, in general, in Section 3.2.3.
We distinguish four binary operations on shapes, namely, sum, product, difference and symmetric difference. First, we define the operations of product, difference and sum, in that order, and prove related properties.

**Definition 2.3** The operation of product, “⋅”: $U_n \times U_n \rightarrow U_n$, computes the common shape of two shapes;
\[
\forall a, b, \varepsilon : \quad \varepsilon \leq a \cdot b \iff \varepsilon \leq a \land \varepsilon \leq b
\]

Figure 2.1 illustrates the product of two shapes in $U_3$.

**Property** The product of two shapes is a part of either shape;
\[
\forall a, b : \quad a \cdot b \leq a \land a \cdot b \leq b \quad \text{[eq2.6]}
\]

**Proof:** Follows directly from $a \cdot b \leq a \cdot b$ and the above definition.

The following statements hold for all $a, b, c$:
\[
a \leq b \iff a \cdot b = a \quad \text{[eq2.7]}
\]
\[
a \cdot 0 = 0 = 0 \cdot a \quad \text{[eq2.8]}
\]
\[
a \leq b \implies a \cdot c \leq b \cdot c \quad \text{[eq2.9]}
\]

**Proof:** [eq2.7]: From the above definition, given that $a \leq a$ and $a \leq b$, it follows that $a \leq a \cdot b$, or $a \cdot b = a$ ([eq2.6] and [eq2.2]). Conversely, $a \cdot b = a \implies a \leq a \cdot b \leq b$.

[eq2.8]: Follows directly from [eq2.7] and the definition of 0.

[eq2.9]: From [eq2.6] and $a \leq b$ it follows that $a \cdot c \leq a \leq b$ and $a \cdot c \leq c$. Then, the above definition specifies that $a \cdot c \leq b \cdot c$. 

Figure 2.1 Two shapes $a$ and $b$ and their product $a \cdot b$. 

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{shapes.png}
\caption{Two shapes $a$ and $b$ and their product $a \cdot b$.}
\end{figure}
Definition 2.4  The operation of difference, “−”: \( U_n \times U_n \to U_n \), computes the relative complement of the first shape with respect to the second shape;
\[
\forall a, b, \varepsilon : \quad \varepsilon \leq a - b \iff \varepsilon \leq a \land \varepsilon \cdot b = 0
\]

Figure 2.2 illustrates the difference of two shapes in \( U_3 \).

Property  The difference of two shapes is a part of the first shape:
\[
\forall a, b : \quad a - b \leq a \quad \text{[eq2.10]}
\]

Proof: Follows directly from \( a - b \leq a - b \) and the above definition.

The following statements hold for all \( a, b, c \):
\[
a - b = a \iff a \cdot b = 0 \quad \text{[eq2.11]}
\]
\[
a - 0 = a \quad \text{[eq2.12]}
\]
\[
a \leq b \implies a - c \leq b - c \quad \text{[eq2.13]}
\]
\[
a \leq b \implies c - b \leq c - a \quad \text{[eq2.14]}
\]

Proof: [eq2.11]: From \( a - b = a \) and the above definition, it follows that \( a \leq a - b \implies a \cdot b = 0 \). Conversely, from \( a \leq a \land a \cdot b = 0 \), it follows that \( a \leq a - b \). We already know that \( a - b \leq a \). Thus, \( a - b = a \).
Algebras of Shapes

[eq2.12]: Follows directly from [eq2.8] and [eq2.11].

[eq2.13]: Given \( a \leq b \), it follows from the above definition and [eq2.4] that
\[
(\forall \epsilon : \epsilon \leq a - c \iff \epsilon \leq b - c \implies \epsilon \leq a - b - c).
\]

[eq2.14]: Given \( a \leq b \), it follows from the above definition and [eq2.9] and [eq2.4] that
\[
(\forall \epsilon : \epsilon \leq c - b \iff \epsilon \leq c \land \epsilon \cdot b = 0 \implies \epsilon \leq c \land \epsilon \cdot a = 0 \iff \epsilon \leq c - a) \implies c - b \leq c - a.
\]

\[\square\]

\textbf{Definition 2.5} The operation of sum, “+”: \( U^n \times U^n \to U^n \), combines two shapes;

\[
\forall a, b, \epsilon : \quad \epsilon \leq a + b \iff \epsilon - b \leq a \land \epsilon - a \leq b
\]

Figure 2.3 illustrates the sum of two shapes in \( U_3 \).

\textbf{Property} Both operands are a part of their sum;

\[
\forall a, b : \quad a \leq a + b \land b \leq a + b \quad [\text{eq2.15}]
\]

\textit{Proof:} From the above definition, given that \( a - b \leq a \) and \( a - a = 0 \leq b \), it follows that \( a \leq a + b \). Idem for \( b \leq a + b \).

\[\square\]

The following statements hold \( \forall a, b, c \):

\[
\begin{align*}
a &\leq b \iff a + b = b \quad [\text{eq2.16}] \\
a + 0 &= a = 0 + a \quad [\text{eq2.17}] \\
a \leq b &\implies a + c \leq b + c \quad [\text{eq2.18}] \\
a + b \leq c &\iff a \leq c \land b \leq c \quad [\text{eq2.19}]
\end{align*}
\]
\subsection*{2.1 Algebraic Model}

\textit{Proof:} [eq2.16]: From the definition of “+” and [eq2.4] it follows that
\[(\forall \varepsilon : \varepsilon \leq a + b \Rightarrow \varepsilon - b \leq a \leq b \Rightarrow \varepsilon \leq b) \Rightarrow a + b \leq b, \text{ or } a + b = b.\]
Conversely, \(a + b = b \Rightarrow a \leq a + b \leq b.\)

[eq2.17]: Follows directly from [eq2.7] and the definition of “0”.

[eq2.18]: The definition of “+” specifies that \(\forall \varepsilon : \varepsilon \leq a + c \Leftrightarrow \varepsilon - a \leq c \land \varepsilon - c \leq a.\) Using \(a \leq b\) and [eq2.14] we derive that \(\varepsilon - b \leq \varepsilon - a \leq c\) and \(\varepsilon - c \leq a \leq b.\) By definition, \(\varepsilon \leq b + c.\) Thus, \(a + c \leq b + c.\)

[eq2.19]: If \(a + b \leq c,\) then \(a \leq a + b \leq c\) and \(b \leq a + b \leq c.\)
Using [eq2.16] and [eq2.18], it follows from \(a \leq c\) and \(b \leq c\) that \(a + b \leq c + b \leq c.\)

\subsection*{2.1.2 Generalized Boolean Algebra}

A partially ordered set, together with operations of union and intersection, defines a lattice. We show that \((U^n, \leq)\) is a relatively complemented distributive lattice under the operations of sum and product, and defines a generalized Boolean algebra \((U^n, (+, \cdot, 0)).\)

\textbf{Definition 2.6} A lattice is a partially ordered set \((L, \leq)\) in which \(x + y\) and \(x \cdot y\) exist for any \(x, y \in L\) (Balbes and Dwinger, 1974).

\textbf{Lemma 2.1} \((U^n, \leq)\) is a lattice.

\textit{Proof:} Follows directly from the definitions of “+” and “−”.

A lattice \((L, \leq),\) together with the operations + and ·, defines an algebra \((L, (+, \cdot)).\)

\textbf{Theorem 2.2} The binary operations of sum and product define an algebra \((U^n, (+, \cdot))\) that satisfies the identities:
\begin{enumerate}
\item \textit{Commutativity of “+”:} \(a + b = b + a.\)
\item \textit{Associativity of “+”:} \(a + (b + c) = (a + b) + c.\)
\item \textit{Idempotency of “+”:} \(a + a = a.\)
\item \textit{Commutativity of “−”:} \(a \cdot b = b \cdot a.\)
\end{enumerate}
Figure 2.4 Three shapes $a$, $b$ and $c$ and their sum $a + b + c$.

(5) Associativity of "•": $a \cdot (b \cdot c) = (a \cdot b) \cdot c$.

(6) Idempotency of "•": $a \cdot a = a$.

(7) Contraction laws: $a + (a \cdot b) = a$ and $a \cdot (a + b) = a$.

(8) $a + b = b \iff a \cdot b = a$.

Proof: Using the anti-symmetric property of "≤", each of the identities $f = g$ is proven by proof of $f \leq g$ and $g \leq f$.

(1) From the definition of "+", [eq2.2] and [eq2.4] it follows that
\[
(\forall \varepsilon : \varepsilon \leq a + b \iff (\varepsilon - b \leq a \land \varepsilon - a \leq b) \iff \varepsilon \leq b + a) \implies a + b = b + a.
\]

(2) The definition of "+" specifies that $\forall \varepsilon : \varepsilon \leq b + c \iff \varepsilon - b \leq c \land \varepsilon - c \leq b$. Using $b \leq a + b$ and [eq2.14] we derive that $\varepsilon - (a + b) \leq \varepsilon - b \leq c$ and $\varepsilon - c \leq b \leq a + b$. By definition, $\varepsilon \leq (a + b) + c$ and, thus, $(b + c) \leq (a + b) + c$. From $a \leq a + b \leq (a + b) + c$ and [eq2.19] it follows that $a + (b + c) \leq (a + b) + c$.

The proof is similar for $(a + b) + c \leq a + (b + c)$.

Figure 2.4 illustrates the associativity of the operation of sum in $U_3$.

(3) From $a \leq a$ and [eq2.16] it follows that $a + a = a$.

(4) From the definition of "•" and [eq2.4] it follows that
\[
(\forall \varepsilon : \varepsilon \leq a \cdot b \iff (\varepsilon \leq a \land \varepsilon \leq b) \iff \varepsilon \leq b \cdot a) \implies a \cdot b = b \cdot a.
\]
2.1 Algebraic Model

(5) From the definition of “•”, [eq2.4] and [eq2.2] it follows that
\((\forall \varepsilon : \varepsilon \leq a \cdot (b \cdot c) \iff (\varepsilon \leq a \land \varepsilon \leq b \land \varepsilon \leq c)\)
\(\iff (\varepsilon \leq a \cdot b \land \varepsilon \leq c) \iff \varepsilon \leq (a \cdot b) \cdot c \Rightarrow a \cdot (b \cdot c) = (a \cdot b) \cdot c.\)

Figure 2.5 illustrates the associativity of the operation of product in \(U_3\).

(6) From \(a \leq a\) and [eq2.7] it follows that \(a \cdot a = a\).

(7) Given \(a \cdot b \leq b\) and [eq2.16] it follows that \(a + (a \cdot b) = (a \cdot b) + a = a\). Similarly, from [eq2.7] and \(a \leq a + b\) it follows that \(a \cdot (a + b) = a\).

(8) [eq2.7] and [eq2.16] specify that \(a + b = b \iff a \leq b \iff a \cdot b = a\).

Theorem 2.2 can also be proven using the notions of least upper bound and greatest lower bound. Suppose \(S\) is a (finite) subset of the partially ordered set \(U_n, S \subseteq U_n\).

**Definition 2.7** An element \(u \in U_n\) is an upper bound for \(S\) if \(a \leq u\) for all \(a \in S\). An upper bound \(u_0\) for \(S\) is a least upper bound for \(S\) if \(u_0 \leq u\) for all upper bounds \(u\) of \(S\). An element \(v \in U_n\) is a lower bound for \(S\) if \(v \leq a\) for all \(a \in S\). A lower bound \(v_0\) for \(S\) is a greatest lower bound for \(S\) if \(v \leq v_0\) for all lower bounds \(v\) of \(S\).

Then, the identities of Theorem 2.2 are all elementary consequences of the above definition (see Balbes and Dwinger, 1974). Conversely, given \((U_n, (+, \cdot))\) an algebra with two binary
operations satisfying the identities of Theorem 2.2, a partial order relation, ≤, can be defined on $U_n$ as follows:

**Corollary 2.3** The relation $\leq$ on $U_n$ defined by $a \leq b \iff a \cdot b = a$, makes $(U_n, \leq)$ into a lattice in which the least upper bound of $a, b$ in $U_n$ is $a + b$ and the greatest lower bound of $a, b$ in $U_n$ is $a \cdot b$.

**Proof:** Following the respective properties of idempotency, commutativity and associativity for “$\cdot$”, the relation $\leq$ is reflexive, antisymmetric and transitive, and therefore a partial order relation. The contraction law $a \cdot (a + b) = a$ implies that $a \leq a + b$, and similarly $b \leq a + b$. If $c$ is an upper bound for $a, b$ then $a \leq c$ and $b \leq c$, or, $a \cdot c = a$ and $b \cdot c = b$. From the identity $a + b = b \iff a \cdot b = a$, it follows that $a + c = c$ and $b + c = c$, so $(a + b) + c = (a + b) + (a + c) = c + c = c$. Therefore, $a + b \leq c$ and $a + b$ is the least upper bound for $a, b$.

Similarly, it is proven that $a \cdot b$ is the greatest lower bound for $a, b$. 

**Property** The zero element, 0, is the least upper bound for $\emptyset \subseteq U_n$.

By Theorem 2.2, we prove that the following statements hold for all $a, b, c$.

\[
a \cdot (b - c) = (a \cdot b) - (a \cdot c) = (a \cdot b) - c \quad [eq.20]
\]

\[
a \leq b \iff a - b = 0 \quad [eq.21]
\]

\[
(a - b) + b = a + b \quad [eq.22]
\]

\[
(a + b) - b = a - b \quad [eq.23]
\]

**Proof:** $[eq.20]$: Firstly, we prove that $a \cdot (b - c) = (a \cdot b) - (a \cdot c)$, using $[eq.2.2]$ and $[eq.2.4]$ and the definitions of “$-$” and “$\cdot$”:

\[
\forall \varepsilon : \varepsilon \leq (a \cdot b) - (a \cdot c) \iff \varepsilon \leq a \cdot b \land \varepsilon \cdot (a \cdot c) = 0
\]

\[
\iff \varepsilon \leq a \land \varepsilon \leq b \land \varepsilon \cdot c = 0 \quad (eq.2.7): \varepsilon \leq a \Rightarrow \varepsilon \cdot a = \varepsilon)
\]

\[
\iff \varepsilon \leq a \land \varepsilon \leq b - c
\]

\[
\iff \varepsilon \leq a \cdot (b - c).
\]

Similarly, we prove that $(a \cdot b) - (a \cdot c) = (a \cdot b) - c$:
2.1 Algebraic Model

\[ \forall \varepsilon : \varepsilon \leq (a \cdot b) - (a \cdot c) \Leftrightarrow \varepsilon \leq a \cdot b \land \varepsilon \cdot (a \cdot c) = 0 \]
\[ \Leftrightarrow \varepsilon \leq a \cdot b \land \varepsilon \cdot c = 0 \]
\[ \Leftrightarrow \varepsilon \leq (a \cdot b) - c. \]

(eq2.21): Using (eq2.7), it follows from \( a - b \leq a \leq b \Rightarrow (a - b) \cdot b = (a - b) \).

However, since \( a - b \leq a - b \), the definition of "\( - \)" specifies that \( (a - b) \cdot b = 0 \). We conclude that \( a - b = 0 \).

Conversely, we prove that \( a - b \neq 0 \) if \( a \notin b \) and \( a \neq 0 \) (\( a = 0 \Rightarrow a \leq b \)). Assume that \( a \cdot b = 0 \). It follows from \( \text{eq2.11} \) that \( a - b = a \neq 0 \). Otherwise, let \( a \cdot b = c \neq 0 \). From \( \text{eq2.20} \), we derive that \((a \cdot b) - a \cdot (a - b) = 0 \). Since \( a \cdot b \leq a \cdot b \), using the result obtained above, we derive that \((a \cdot b) - a \cdot (a - b) = 0 \), or \( b \cdot (a - c) = 0 \). Again, it follows from \( \text{eq2.11} \) that \( (a - c) - b = (a - c) \neq 0 \). Given that \( a - c \leq a \), and therefore \((a - c) - b \leq a - b \) (eq2.13), we conclude that \( a - b \neq 0 \).

(eq2.22): From \( b \leq a + b \) and \( a - b \leq a \leq a + b \) and using \( \text{eq2.19} \), it follows that \( (a \cdot b) + b \leq a + b \). We know that \( b \leq (a - b) + b \). We now prove that \( a \leq (a - b) + b \). Otherwise, suppose \( \exists \varepsilon \neq 0 : \varepsilon \leq a \land \varepsilon \cdot (a - b) + b = 0 \Leftrightarrow \varepsilon \leq a \land \varepsilon \cdot (a - b) = 0 \land \varepsilon \cdot b = 0 \). Using \( \text{eq2.20} \) and \( \text{eq2.21} \) we derive that \( \varepsilon \cdot (a - b) = (\varepsilon \cdot a) - b = \varepsilon - b = 0 \Leftrightarrow \varepsilon \leq b \).

However we have that \( \varepsilon \cdot b = 0 \) and, therefore, \( \varepsilon = 0 \), which violates our assumption.

(eq2.23): \( \forall \varepsilon : \varepsilon \leq (a + b) - b \Leftrightarrow \varepsilon \leq a + b \land \varepsilon \cdot b = 0 \)
\[ \Leftrightarrow \varepsilon - a \leq b \land \varepsilon - b \leq a \land \varepsilon \cdot b = 0 \]
\[ \Leftrightarrow \varepsilon - a \leq b \land \varepsilon \leq a \land \varepsilon \cdot b = 0 \] \text{([eq2.11])}
\[ \Leftrightarrow 0 \leq b \land \varepsilon \leq a \land \varepsilon \cdot b = 0 \] \text{([eq2.21])}
\[ \Leftrightarrow \varepsilon \leq a - b. \]

We now prove that \( (U_{\mu}, \leq) \) is a distributive lattice.

**Definition 2.8** A lattice \((L, \leq)\) is **distributive** if it satisfies one of the following two identities:

\( \forall x, y, z \in L : \)
\( x \cdot (y + z) = (x \cdot y) + (x \cdot z) \)
\( \forall x, y, z \in L : \)
\( x + (y \cdot z) = (x + y) \cdot (x + z) \)
Property The operation of product distributes over the operation of sum;
\[ \forall a, b : \quad a \cdot (b + c) = (a \cdot b) + (a \cdot c) \] \{eq2.24\}

Proof: From the definitions of “⋅” and “+” it follows that
\[ \forall \varepsilon : \varepsilon \leq a \cdot (b + c) \iff \varepsilon \leq a \land \varepsilon \leq b + c \iff \varepsilon \leq a \land \varepsilon - c \leq b \land \varepsilon - b \leq c. \]
Given that \( \varepsilon - b \leq \varepsilon \leq a \) and \( \varepsilon - b \leq c \), we derive that \( \varepsilon - b \leq a \cdot b \) and, similarly, \( \varepsilon - c \leq a \cdot b \).
Also, using [eq2.20] \( \varepsilon \leq a \Rightarrow a \cdot \varepsilon = \varepsilon \Rightarrow \varepsilon - (a \cdot b) = (a \cdot \varepsilon) - (a \cdot b) = a \cdot (\varepsilon - b) \leq \varepsilon - b \) and, similarly, \( \varepsilon - (a \cdot c) \leq \varepsilon - c. \) The definition of “+” specifies that \( \varepsilon \leq (a \cdot b) + (a \cdot c) \) and, thus, \( a \cdot (b + c) \leq (a \cdot b) + (a \cdot c). \)
Since \( b \leq b + c \), it follows from [eq2.9] that \( a \cdot b \leq a \cdot (b + c) \); similarly, \( a \cdot c \leq a \cdot (b + c). \)
Thus, \( (a \cdot b) + (a \cdot c) \leq a \cdot (b + c). \) 

Figure 2.6 illustrates the distributivity of “⋅” over “+” in \( U_3 \).

Property The operation of sum distributes over the operation of product;
\[ \forall a, b : \quad a + (b \cdot c) = (a + b) \cdot (a + c) \] \{eq2.25\}

Proof: \( (a + b) \cdot (a + c) = ((a + b) \cdot a) + ((a + b) \cdot c) = a + (a \cdot c) + (b \cdot c) = a + (b \cdot c) \)

Corollary 2.4 \((U_n, \leq)\) is a distributive lattice.

Proof: Follows directly from Lemma 2.1 and [eq2.24] and [eq2.25].

The operation of difference defines a relative complement for each shape, resulting in a relatively complemented distributive lattice \((U_n, \leq)\).

Definition 2.9 Let \((L, \leq)\) be a lattice. An element \( x \in L \) is relatively complemented if
\[ \forall u, v \in L : u \leq x \leq v \Rightarrow \exists y \in L, u \leq y \leq v : x \cdot y = u \land x + y = v. \]

Definition 2.10 A relatively complemented distributive lattice is a distributive lattice such that every element is relatively complemented.

Lemma 2.5 \((U_n, \leq)\) is a relatively complemented distributive lattice.
2.1 Algebraic Model

Proof: \(\forall a, b, x \in U_n : a \leq x \leq b\), consider \(y = (b - x) + a \in U_n\). From \(a \leq b\), [eq2.18] and [eq2.16], it follows that \(y = (b - x) + a \leq b + a = b\). Also, \(a \leq (b - x) + a \leq y\) and, thus, \(a \leq y \leq b\). Using [eq2.24], [eq2.20], \(a \leq x\) and [eq2.14], we derive that \(x \cdot y = x \cdot ((b - x) + a) = (x \cdot (b - x)) + (x \cdot a) = ((x \cdot b) - x) + a = 0 + a = a\).

Similarly, from [eq2.22] and \(a \leq x \leq b\), it follows that \(x + y = x + (b - x) + a = x + (b - x) = x + b = b\).

\[\square\]

Definition 2.11 A relatively complemented distributive lattice \((L, \leq)\) with a zero element defines a generalized Boolean algebra \((L, (+, \cdot, 0))\).
Corollary 2.6 \((\mathbb{U}_n, (+, \cdot, 0))\) is a generalized Boolean algebra and the difference, \(a \cdot b\), \(a, b \in \mathbb{U}_n\) is the relative complement of \(a \cdot b\) for \(0 \leq a \cdot b \leq a\).

Proof: Follows directly from Lemma 2.5 and its proof.

Figure 2.7 illustrates the relative complement of a shape in \(U_3\) under the operation of difference.

2.1.3 Boolean Ring

Using the operations of product and symmetric difference, the generalized Boolean algebra \((\mathbb{U}_n, (+, \cdot, 0))\) becomes a Boolean ring \((\mathbb{U}_n, (\oplus, \cdot, 0))\), with “\(\oplus\)” the operation of symmetric difference. We remark that the operations of product and difference may be used to define a partitioning on one or more shapes.

Lemma 2.7 The operations of product and difference define a partitioning on a shape with respect to another shape. That is,
\[
\forall a, b : \quad a = (a - b) + (a \cdot b) \land (a - b) \cdot (a \cdot b) = 0
\]
Proof: We prove that \( a \leq (a - b) + (a \cdot b) \) by negation:

If \( a \not\leq (a - b) + (a \cdot b) \) then \( \exists \varepsilon \neq 0 : \varepsilon \leq a \wedge \varepsilon \cdot ((a - b) + (a \cdot b)) = 0 \)

\[
\begin{align*}
\Rightarrow & \quad \varepsilon \leq a \wedge \varepsilon \cdot (a - b) + \varepsilon \cdot (a \cdot b) = 0 \\
\Rightarrow & \quad \varepsilon \leq a \wedge \varepsilon \cdot (a - b) = 0 \wedge \varepsilon \cdot (a \cdot b) = 0 \\
\Rightarrow & \quad \varepsilon \leq a \wedge (\varepsilon \cdot a - b) = 0 \wedge (\varepsilon \cdot a \cdot b) = 0 \\
\Rightarrow & \quad \varepsilon \leq a \wedge \varepsilon \cdot a \leq b \wedge (\varepsilon \cdot a \cdot b) = 0 \\
\Rightarrow & \quad \varepsilon \leq a \wedge \varepsilon \cdot a = 0 \\
\Rightarrow & \quad \varepsilon = 0, \text{ which violates our assumption.}
\end{align*}
\]

Therefore, \( a \leq (a - b) + (a \cdot b) \). Furthermore, we have that \( a - b \leq a \) and \( a \cdot b \leq a \), from which it follows that \( (a - b) + (a \cdot b) \leq a \) ([eq2.18]). This completes the proof of \( a = (a - b) + (a \cdot b) \).

Finally, using [eq2.20] and [eq2.7], \( (a \cdot b) \cdot (a - b) = ((a \cdot b) \cdot a) - b = (a \cdot b) - b = 0 \). \( \Box \)

---

**Lemma 2.8** The operations of product and difference define a partitioning on the sum of two shapes. That is,

\[
\begin{align*}
& a + b = (a - b) + (b - a) + (a \cdot b) \wedge \\
& (a - b) \cdot (b - a) = 0 \wedge (a - b) \cdot (a \cdot b) = 0 \wedge (a \cdot b) \cdot (b - a) = 0
\end{align*}
\]

Proof: We know from Lemma 2.7 that \( a = (a - b) + (a \cdot b) \) and, similarly, \( b = (b - a) + (b \cdot a) \). It follows that \( a + b = (a - b) + (a \cdot b) + (b - a) + (b \cdot a) = (a - b) + (b - a) + (a \cdot b) + (b \cdot a) = (a - b) + (b - a) + (a \cdot b) + (b \cdot a) \), because \( a \cdot b = b \cdot a \). Lemma 2.7 also specifies that \( (a - b) \cdot (a \cdot b) = 0 \) and \( (a \cdot b) \cdot (b - a) = 0 \). Using [eq2.20] and \( a - b \leq a - b \Rightarrow (a - b) \cdot b = 0 \), we prove that \( (a - b) \cdot (b - a) = ((a - b) \cdot b) - a = 0 - a = 0 \). \( \Box \)

Figure 2.8 illustrates the partitioning of two shapes in \( U_3 \) with respect to each other.

The partitioning of shapes leads naturally to the definition of the operation of symmetric difference.

**Definition 2.12** The operation of symmetric difference, \( \bigoplus \): \( U_n \times U_n \rightarrow U_n \), computes the sum of the differences of both shapes;
Property The symmetric difference of two shapes equals the difference of their sum and product;
\[ \forall a, b : \quad a \oplus b = (a + b) - (a \cdot b) \] [eq2.26]

Proof: From Lemma 2.8 and [eq2.24] we derive that
\[(a - b) + (b - a) \cdot (a \cdot b) = ((a - b) \cdot (a \cdot b)) + ((b - a) \cdot (a \cdot b)) = 0 + 0 = 0.\]
Thus, \((a + b) - (a \cdot b) = ((a - b) + (b - a) + (a \cdot b)) - (a \cdot b)\)
\[= ((a - b) + (b - a)) - (a \cdot b)\]  
\[= (a - b) + (b - a).\]  
\[\text{(eq2.23)}\]

\[= ((a - b) + (b - a)) - (a \cdot b)\]  
\[= (a - b) + (b - a).\]  
\[\text{(eq2.11)}\]

\[\Box\]

**Property**

\[\forall a, b : \quad a = (a - b) \oplus (a \cdot b)\]  
\[\text{(eq2.27)}\]

**Proof:** Given [eq2.11] and \((a - b) \cdot (a \cdot b) = 0\), it follows from the above definition that
\[a \oplus b = (a - b) - (a \cdot b) = (a - b) + (a \cdot b) = a.\]  
\[\Box\]

The following statements hold for all \(a, b, c:\)

\[a \leq b \iff a \oplus b = b - a\]  
\[\text{(eq2.28)}\]

\[a \oplus 0 = a = 0 \oplus a\]  
\[\text{(eq2.29)}\]

\[a \oplus a = 0\]  
\[\text{(eq2.30)}\]

**Proof:** [eq2.28]: From the above definition and [eq2.16], it follows that
\[a \oplus b = (a - b) + (b - a) = b - a \iff a \leq b.\]  
\[\Box\]

[eq2.29] and [eq2.30]: Follow directly from [eq2.28].

**Proof:** [eq2.28]: From the above definition and [eq2.16], it follows that
\[a \oplus b = (a - b) + (b - a) = b - a \iff a \leq b.\]  
\[\Box\]

**Proof:** [eq2.28]: From the above definition and [eq2.16], it follows that
\[a \oplus b = (a - b) + (b - a) = b - a \iff a \leq b.\]  
\[\Box\]

Given two shapes \(a\) and \(b\), consider the set \(S = \{a - b, b - a, a \cdot b\}\) of classes from the partitioning specified in Lemma 2.8. Then, there is a one-to-one correspondence between the subsets of \(S\), i.e., the elements in the power set of \(S\), and the shapes 0, \(a\) and \(b\) and the shapes derived from these using the operations of sum, product, difference and symmetric difference:

\[
\begin{align*}
\{\} & \leftrightarrow 0 \\
\{a - b\} & \leftrightarrow a - b \\
\{b - a\} & \leftrightarrow b - a \\
\{a \cdot b\} & \leftrightarrow a \cdot b \\
\{a - b, b - a, a \cdot b\} & \leftrightarrow a + b \\
\{a - b, a \cdot b\} & \leftrightarrow a \\
\{b - a, a \cdot b\} & \leftrightarrow b \\
\{a - b, b - a\} & \leftrightarrow a \oplus b
\end{align*}
\]

Figure 2.10 illustrates, for two shapes in \(U_3\), all seven shapes corresponding to non-empty subsets of \(S\). Figure 2.11 shows a graphical description of the corresponding lattice or partially ordered set. The levels in the “graph” correspond to subsets of \(S\) with the same
cardinality. Balbes and Dwinger (1974) prove that every partially ordered set is isomorphic with a set of subsets of some set.

$U_n$ constitutes an Abelian group under the operation of symmetric difference.

**Definition 2.13** An Abelian group consists of a non-empty set $L$ and a binary operation $\oplus$ with the following properties, $\forall a, b, c \in L$:

1. Closure: $a \oplus b$ is a unique element of $L$.
2. Associativity: $a \oplus (b \oplus c) = (a \oplus b) \oplus c$.
3. Identity: $\exists e \in L, \forall a \in L: a \oplus e = a = e \oplus a$.
4. Inverses: $\forall a \in L, \exists a^{-1} \in L: a \oplus a^{-1} = e = a^{-1} \oplus a$.
5. Commutativity: $a \oplus b = b \oplus a$.
The following statements hold \( \forall a, b, c \):

\[
\begin{align*}
    a - (b - c) &= (a - b) + (a \cdot c) \tag{eq2.31} \\
    (a + b) - c &= (a - c) + (b - c) \tag{eq2.32} \\
    a - (b + c) &= (a - b) - c \tag{eq2.33}
\end{align*}
\]

Proof: [eq2.31]: First, we prove the equation in the case that \( a \leq b \);

\[
\forall \varepsilon : \varepsilon \leq a - (b - c) \iff \varepsilon \leq a \land \varepsilon \cdot (b - c) = 0 \iff \varepsilon \leq a \land (\varepsilon \cdot b) - c = 0
\]

\[
\iff \varepsilon \leq a \leq b \land \varepsilon \cdot b = \varepsilon \leq c \iff \varepsilon \leq a \cdot c.
\]

It follows that \( a - (b - c) = a \cdot c = (a - b) + (a \cdot c) \).

Now, we prove the equation for arbitrary \( a \) and \( b \);

\[
\forall \varepsilon : \varepsilon \leq (a - b) + (a \cdot c) \iff \varepsilon - (a \cdot c) \leq a - b \land \varepsilon - (a - b) \leq a \cdot c
\]

\[
\iff \varepsilon - (a \cdot c) \leq a \land (\varepsilon - (a \cdot c)) \cdot b = 0 \land \varepsilon - (a - b) \leq a \land \varepsilon - (a - b) \leq c.
\]

From \( \varepsilon - (a \cdot c) = (\varepsilon - a) + (\varepsilon - c) \leq a \), it follows that \( \varepsilon - a \leq a \iff (\varepsilon - a) \cdot a = (\varepsilon \cdot a) - a = 0 = \varepsilon - a \) or \( \varepsilon \leq a \). Conversely, if \( \varepsilon \leq a \) then \( \varepsilon - (a \cdot c) \leq a \) and \( \varepsilon - (a - b) \leq a \). Furthermore, we have that \( (\varepsilon - (a \cdot c)) \cdot b = 0 \iff (\varepsilon \cdot b) - (a \cdot c) = 0 \iff \varepsilon \cdot b \leq a \cdot c \) and, given the case that \( \varepsilon \leq a, \varepsilon - (a - b) = \varepsilon \cdot b \leq a \cdot c \leq c \).
Thus, \( \forall \varepsilon : \varepsilon \leq (a - b) + (a \cdot c) \iff \varepsilon \leq a \land \varepsilon \cdot b \leq c \iff \varepsilon \leq a \land (\varepsilon \cdot b) - c = 0 \)
\( \iff \varepsilon \leq a \land \varepsilon \cdot (b - c) = 0 \iff \varepsilon \leq a - (b - c). \)

[eq2.32]: \( \forall \varepsilon : \varepsilon \leq (a - c) + (b - c) \iff \varepsilon - (a - c) \leq b - c \land \varepsilon - (b - c) \leq a - c \)
\( \iff (\varepsilon - a) + \varepsilon \cdot c \leq b - c \land (\varepsilon - b) + \varepsilon \cdot c \leq a - c \)
\( \iff \varepsilon - a \leq b - c \land \varepsilon \cdot c \leq b - c \land \varepsilon - b \leq a - c \land \varepsilon \cdot c \leq a - c \)
\( \iff \varepsilon - a \leq b \land (\varepsilon - a) \cdot c = 0 \land \varepsilon \cdot c \leq b \land (\varepsilon \cdot c) \cdot c = 0 \land \)
\( \iff \varepsilon - b \leq a \land (\varepsilon - b) \cdot c = 0 \land \varepsilon \cdot c \leq a \land (\varepsilon \cdot c) \cdot c = 0 \)
\( \iff \varepsilon - a \leq b \land \varepsilon - b \leq a \land \varepsilon \cdot c = 0 \)
\( \iff \varepsilon \leq a + b \land \varepsilon \cdot c = 0 \)
\( \iff \varepsilon \leq (a + b) - c, \) given that
\( (\varepsilon \cdot c) \cdot c = 0 \iff \varepsilon \cdot c = 0 \iff (\varepsilon - a) \cdot c = 0 \land (\varepsilon - b) \cdot c = 0. \)

[eq2.33]: \( \forall \varepsilon : \varepsilon \leq a - (b + c) \iff \varepsilon \leq a \land \varepsilon \cdot (b + c) = 0 \)
\( \iff \varepsilon \leq a \land \varepsilon \cdot b = 0 \land \varepsilon \cdot c = 0 \iff \varepsilon \leq a - b \land \varepsilon \cdot c = 0 \iff \varepsilon \leq (a - b) - c \)

\[ \textbf{Lemma 2.9} \] The algebra \( U_n \) satisfies the axioms of an Abelian group under symmetric difference (with identity 0 and inverse \( a^{-1} = a \)).

\[ \textbf{Proof}: \] \( a, b \) and \( c \) are any three elements of \( U_n \).

1. The definition of the operation of symmetric difference specifies that \( a \oplus b \in U_n \), and the resulting shape is unique.

2. \( a \oplus (b \oplus c) = (a - (b \oplus c)) + ((b \oplus c) - a) \)
\[ = (a - ((b + c) - (b \cdot c))) + ((b - c) + (c - b)) - a \]  
\[ = (a - (b + c)) + (a - (b \cdot c)) + ((b - c) - a) + ((c - b) - a) \]  
\[ \text{([eq2.26])} \]
\[ = (a - (b + c)) + (a \cdot b) + (b - (a + c)) + (c - (a + b)) \]  
\[ \text{([eq2.31] and [eq2.32])} \]
\[ = (a - (b + c)) + (b - (a + c)) + (c - (a + b)) + (c \cdot a \cdot b) \]
\[ = ((a - b) - c) + ((b - a) - c) + (a \cdot (a - b) + (a \cdot b)) \]
\[ = ((a - b) + (b - a)) - c + (c - ((a + b) - (a \cdot b))) \]
\[ = ((a \oplus b) - c) + (c - (a \oplus b)) \]
\[ = (a \oplus b) \oplus c \]
2.1 Algebraic Model

Figure 2.12 Three shapes $a$, $b$ and $c$ and their symmetric difference $a \oplus b \oplus c$.

Figure 2.12 illustrates the associativity of the operation of symmetric difference in $U_3$.

(3) The empty shape is the identity: $a \oplus 0 = a = 0 \oplus a$.

(4) The inverse of any shape is the shape itself: $a \oplus a = 0 = a \oplus a$.

(5) $a \oplus b = (a - b) + (b - a) = (b - a) + (a - b) = b \oplus a$. □

If $(U_n, (+, \cdot, 0))$ is a generalized Boolean algebra, then $(U_n, (\oplus, \cdot, 0))$ is a Boolean ring.

**Definition 2.14** A Boolean ring is a non-empty set $L$ in which two binary operations $\oplus$ and $\cdot$ are defined satisfying the following conditions (Arnold, 1962), $\forall a, b, c \in L$:

1. **Closure**: $a \oplus b$ and $a \cdot b$ are unique elements of $L$.
2. **Commutativity of “$\oplus$”**: $a \oplus b = b \oplus a$.
3. **Associativity of “$\oplus$”**: $a \oplus (b \oplus c) = (a \oplus b) \oplus c$.
4. **Solvability of equations**: The equation $a \oplus x = b$ has at least one solution in $L$.
5. **Associativity of “$\cdot$”**: $a \cdot (b \cdot c) = (a \cdot b) \cdot c$.
6. **Distributivity**: $a \cdot (b \oplus c) = (a \cdot b) \oplus (a \cdot c)$ and $(a \oplus b) \cdot c = (a \cdot c) \oplus (b \cdot c)$.
7. **Idempotency of “$\cdot$”**: $a \cdot a = a$. 
Theorem 2.10  The algebra $U_n$ satisfies the axioms of a Boolean ring under symmetric difference and product.

Proof: $a$, $b$ and $c$ are any three elements of $U_n$.

(1) The definitions of the operations of product and symmetric difference specify that $a \oplus b \in U_n$ and $a \cdot b \in U_n$, and the resulting shapes are unique.

(2) See Lemma 2.9.

(3) See Lemma 2.9.

(4) The equation $a \oplus x = b$ has at least one solution in $U_n$: $x = a \oplus b$, since

$$a \oplus (a \oplus b) = (a \oplus a) \oplus b = 0 \oplus b = b.$$  

(5) See Theorem 2.2.

(6) $a \cdot (b \oplus c) = a \cdot ((b - c) + (c - b))$

\[= (a \cdot (b - c)) + (a \cdot (c - b)) \quad ([eq2.24]) \]

\[= ((a \cdot b) - (a \cdot c)) + ((a \cdot c) - (a \cdot b)) \quad ([eq2.20]) \]

\[= (a \cdot b) \oplus (a \cdot c). \]

$(a \oplus b) \cdot c = c \cdot (a \oplus b) = (c \cdot a) \oplus (b \cdot a) = (a \cdot c) \oplus (b \cdot c)$.

(7) See Theorem 2.2.  

When given that $(U_n, (\oplus, \cdot, 0))$ is a Boolean ring, the operations of sum and difference may be defined as $a - b = a \cdot (a \oplus b)$ and $a + b = a \oplus b \oplus (a \cdot b)$, and $(U_n, (+, \cdot, 0))$ is a generalized Boolean algebra. We prove these equations as follows:

$$a \cdot (a \oplus b) = a \cdot ((a - b) + (b - a))$$

\[= (a \cdot (a - b)) + (a \cdot (b - a)) \]

\[= ((a \cdot b) - b) + ((a \cdot b) - a) \]

\[= (a - b) + 0 \]

\[= a - b. \]
2.2 Shape Relations

We consider the following relations among shapes in $U_n$, namely, contain, overlap and disjoint (as well as equality). The relation contain has been introduced in Section 2.1.1.

**Definition 2.15** A shape $a$ contains a shape $b$ if $b$ is a part of $a$, $b \leq a$.

If a shape $a$ contains a shape $b$, then the difference of $b$ and $a$ equals zero, $b - a = 0$. Similarly, if the product of two shapes equals zero, the shapes are said to be disjoint. Note that two shapes are equal only if the symmetric difference of the two shapes is zero.

**Definition 2.16** Two shapes $a$ and $b$ are disjoint if their product is zero, $a \cdot b = 0$.

If two shapes are neither disjoint, nor one contains the other, then, they are said to overlap.

**Definition 2.17** Two shapes $a$ and $b$ overlap if their product is non-zero and neither shape contains the other, $a \cdot b \neq 0 \land a \not\subseteq b \land b \not\subseteq a$.

These relations are mutually exclusive and exactly one relation applies to any pair of shapes, within the same algebra. The relation contain is reflexive, anti-symmetric and transitive.

Similarly, given that $(U_n, (+, \cdot, 0))$ is a generalized Boolean algebra, the operation of difference is defined as $a - b = c \iff b \cdot c = 0 \land b + c = a + b$, while the definition of the operation of symmetric difference remains the same.

The proof consists of two parts: Firstly, assume $a - b = c$. The definition of “$-$” specifies that $b \leq a$ and $c \cdot b = 0$. Using [eq2.18], it follows that $c + b = (a - b) + b = a + b$. Secondly, assume $b \cdot c = 0$ and $b + c = a + b$. Then, $a - b = (a + b) - b = (c + b) - b = c - b = c$.

Finally, we can define the operations of sum and product, using the operations of difference and symmetric difference, as $a \cdot b = a - (a - b)$ and $a + b = a \oplus b \oplus (a - (a - b))$.

2.2 Shape Relations

We consider the following relations among shapes in $U_n$, namely, contain, overlap and disjoint (as well as equality). The relation contain has been introduced in Section 2.1.1.

**Definition 2.15** A shape $a$ contains a shape $b$ if $b$ is a part of $a$, $b \leq a$.

If a shape $a$ contains a shape $b$, then the difference of $b$ and $a$ equals zero, $b - a = 0$. Similarly, if the product of two shapes equals zero, the shapes are said to be disjoint. Note that two shapes are equal only if the symmetric difference of the two shapes is zero.

**Definition 2.16** Two shapes $a$ and $b$ are disjoint if their product is zero, $a \cdot b = 0$.

If two shapes are neither disjoint, nor one contains the other, then, they are said to overlap.

**Definition 2.17** Two shapes $a$ and $b$ overlap if their product is non-zero and neither shape contains the other, $a \cdot b \neq 0 \land a \not\subseteq b \land b \not\subseteq a$.

These relations are mutually exclusive and exactly one relation applies to any pair of shapes, within the same algebra. The relation contain is reflexive, anti-symmetric and transitive.
The relations overlap and disjoint are anti-reflexive, symmetric and non-transitive. Figure 2.13 illustrates these relations on pairs of shapes in $U_3$.

**Figure 2.13** The shape relations (a) contain, (b) overlap and (c, d) disjoint.
Chapter 3
Geometry of Shapes

The algebraic model for shapes, laid out in Chapter 2, is independent of an underlying geometric framework. We distinguish the geometry of shapes from Euclidean geometry, instead, consider a mapping to Euclidean space in order to realize the different algebra’s $U_{n,k}$ within a single space $E^k$. We consider a canonical representation for shapes as sets of (maximal) segments, where line, plane and volume segments are defined as elements of different algebras $U_{n,k}$ ($n = 1$ to $3$) with a mapping from $U_{n,k}$ to $U_{n-1,k}$, as well as from $U_{n,k}$ to the power set of the $k$-dimensional Euclidean space, $\wp(E^k)$. Similarly, we define an isomorph mapping from $U_{0,k}$ to the Euclidean space $E^k$. The segments that compose a shape can be partitioned into co-equal equivalence classes based on the hyperplanes in $E^k$ that carry them. This carrier relation is represented by the co-descriptor of the segment, which expresses a mapping from $U_{n,k}$ to $E^k$. Then, two segments in the same algebra can combine only if they belong to the same equivalence class; thus, if they are co-equal. The boundary of a segment or shape is a mapping from $U_{n,k}$ to $U_{n-1,k}$ such that the boundaries

1. In Chapter 4, we consider an isomorphism between $U_{n,k}$ and a subset of $\wp(E^k)$.
2. Co-equality is defined in Section 3.1.1.
of the shape resulting from each of the operations of sum, product, difference and symmetric difference form a part of the sum of the boundaries of both operands. We say that two co-equal segments share boundary if they do not overlap, nor one contains the other, but their boundaries overlap; two shapes that do not overlap, nor share boundary, and of which one does not contain the other, are said to be disjoint. A segment of a shape in \( U_{n,k} \) that is disjoint with respect to all other segments in the shape, and the boundary of which constitutes a maximal shape in \( U_{n-1,k} \), is denoted a maximal segment. A shape is said to be maximal if it consists only of maximal segments; its representation is termed a maximal representation. Table 3.1 summarizes the mappings for each algebra.

In the sequel, we consider algebras \( U_{n,k} \) in a \( k \)-dimensional space, in general, and algebras \( U_{n,3} \) in a 3-dimensional space, in particular. When appropriate we use \( U_n \) to denote \( U_{n,k} \).

### 3.1 Shape Representation

The representation of shapes is an algebra-specific mapping from \( U_n \) to \( U_{n-1} \times \wp(\mathbb{E}^k) \) except for shapes of points, for which there exists an isomorphism between \( U_0 \) and \( \mathbb{E}^k \).

#### 3.1.1 Co-descriptor

The basic entity in \( U_0 \) is the point. A point in \( U_0 \) is isomorphic to a point in \( \mathbb{E}^k \) that is represented by \( k \) independent coordinates with respect to the defined axes. Given a point \( p \)
in $U_0$, we denote the isomorph point, $p^*$ in $E_k$, the representation of $p$. We define a total order on the points in $U_0$ corresponding to the lexicographical order on the coordinates of the isomorph points in $E_k$. We denote the order relation $\leq_c$ and, given two points $p$ and $q$, we write $p \leq_c q$ if $p^*$ is lexicographically less than or equal to $q^*$, and $q \leq_c p$ otherwise.

A point is embedded in another point if their representations are identical and, therefore, the points are identical.

**Definition 3.1** A shape of points is a finite set of points. A shape of points is embedded in another shape in $U_0$ if each point in the first shape is identical to a point in the second shape.

The basic entity in $U_1$ is the line segment, in $U_2$ the plane segment, and in $U_3$ the volume segment. A segment is characterized by a (unique) co-descriptor as well as a boundary. Any segment is a shape, but a shape is not necessarily a segment, yet can be considered as composed, under the operation of sum or symmetric difference, of a finite set of segments from the same algebra.

Consider the set of infinite $n$-dimensional hyperplanes in $E_k$. The co-descriptor expresses a mapping from $U_n$ to $E_k$ that maps every segment $s$ in $U_n$ to a hyperplane in $E_k$ that is denoted the carrier of the segment, and also, $\text{co}[s]$. Two segments are said to be co-equal if they have the same co-descriptor. (If a segment is a part of another segment, the two segments are necessarily co-equal.) The co-equality relation is reflexive, symmetric and transitive; it is an equivalence relation that partitions any set of segments from $U_n$ into co-equal classes.

We extend the notion of co-equality from segments to shapes: consider a shape as a finite set of segments (from the same algebra).

**Definition 3.2** A shape is co-equal if all segments in the shape have the same co-descriptor.

---

3. The precise definition of a segment is given in Section 3.1.3. Here, it suffices to consider a segment as a shape with a unique co-descriptor. Since this is a less restrictive definition of a segment, all results obtained here remain valid for “proper” segments.

4. We use square brackets for the arguments of certain mappings, e.g., $\text{co}[s]$, in order to emphasize that the result is not so much the image of a function applied to the argument, but an integral part of the representation of this argument.
We assign to a co-equal shape $a$ the unique co-descriptor, denoted $co[a]$. The co-equality relation defines a classification on the segments of a shape into co-equal classes, each of which constitutes a co-equal shape. Two shapes are said to be co-equal if either shape is co-equal and their co-descriptors are identical. Note that there exists only a single infinite 3-dimensional hyperplane in $E^3$. Therefore, all volume segments and shapes in $U_{3,3}$ are necessarily co-equal.

For the sake of uniformity, we consider a point to be a segment in $U_0$, denote the isomorph point in $E^k$ the co-descriptor of the segment (the segment has no boundary) and define two points to be co-equal if they are identical.

### 3.1.2 Boundary

In this section we introduce a boundary as a mapping from $U_n$ to $U_{n-1}$ and show that there exists a relation between the boundaries of two shapes and the boundary of the shape resulting from one of the operations from among sum, product, difference and symmetric difference. In particular, we prove that the boundary of any of the shapes resulting from the above operations is a part of the sum of the boundaries of both original shapes. In the following, we deal with shapes in $U_{n-1}$ that are parts of boundaries of shapes in $U_n$. The findings apply as well, but are not restricted, to segments.

The boundary, shortly denoted as $B$, is an algebra-specific mapping of each segment or shape in $U_n$ onto a shape in $U_{n-1}$. The boundary of a shape $a$ is denoted $B[a]$. No boundary exists for a shape in $U_0$. The boundary of the zero element in $U_n$ ($n \geq 1$) is the zero element in $U_{n-1}$: $B[0] = 0$.

We define a neighborhood\(^5\) $\Delta(s)$ of a shape $s$ as a shape that has $s$ as a part of its boundary. Note that any shape in $U_{n+1}$ is a neighborhood for the zero element in $U_n$, in particular, 0.

---

\(^5\) This definition serves two purposes: Even though it is never specifically required as such, we adopt the connotation for a neighborhood of a small and proximate shape. A neighborhood also implicitly specifies a boundary condition on a shape, removing the need for such a condition to be repeated explicitly.
Definition 3.3 A neighborhood $\Delta(s)$ of a shape $s$ in $U_n$ is a shape in $U_{n+1}$ that has $s$ as a part of its boundary;

$$\forall s \in U_n, \exists \Delta(s) \in U_{n+1}: s \leq B[\Delta(s)].$$

The following assertions serve the definition of the boundary shape of a shape. They specify the existence of disjoint neighborhoods that characterize a shape as a part of a boundary as well as the behavior of the neighborhood relation in the presence of another shape.\(^6\) We construct the proofs to these assertions using an isomorphism between shapes in a $k$-dimensional space and point sets in the Euclidean space $E^k$ (see Section 4.2.2)\(^7\), and elementary concepts from point set topology (Henle, 1979).

Consider a shape $a$ in $U_{n,k}$ and a segment $s$ in $U_{n-1,k}$ that is a part of the boundary of $a$. Let $H$ denote the $n$-dimensional hyperplane in $E^k$ that is the carrier of $a$. Under the isomorphism, $a$ defines a point set $A$ that is a subset of $H$. Similarly, the segment $s$ defines a point set $S$ that is a subset of the boundary of $A$. Define $H$ to be a topological space such that a (point set) neighborhood of a point $p$ in $H$ is given by $\delta_{\text{neighborhood}}(p) = \{ q \mid d(p,q) < \delta \}$, under the Euclidean distance defined over $H$. A point $p$ is considered near a point set if every neighborhood of $p$ contains a point of $P$. We denote $P^\perp$ as the complementary point set, within $H$, of a point set $P$, i.e., $P^\perp = H \setminus P$. The closure of a point set $P$, denoted $c(P)$, is the set of points near $P$; the interior of a point set $P$, denoted $i(P)$, is the set of points not near $P$; the boundary of a point set $P$, denoted $b(P)$, is the set of points both near $P$ and near $P^\perp$. Given an $n$-dimensional point set $P$, $r(P)$ denotes the $n$-regularized point set, conform to the isomorphism.

We can distinguish the equivalent of two disjoint sides (left and right) to a point, line or plane, within a hyperplane in $E^3$, to a segment in $U_n$.

\(^6\) Even though we specify these assertions for (rectilinear) shapes and segments, equivalent versions can be asserted and proven that hold for curved shapes as well.

\(^7\) The isomorphisms between $U_{n,k}$ and a subset of $\mathcal{P}(E^k)$ are only narrowly proven in Section 4.2.2: a unique correspondence between shapes and point sets is shown, that holds under the respective boundary relations. The equivalence between the arithmetic shape operations and the Boolean set operations, as well as between the part relation on shapes and the subset relation on point sets, is shown (implicitly) in Chapter 5.
In $\mathbb{E}^3$, consider an infinite plane $s$ with equation $ax + by + cz = d$. Then, the plane $s$ divides the space $\mathbb{E}^3$ in two spatial (and one planar) regions defined by the set of points $(x, y, z)$ for which the value of $ax + by + cz$ is, respectively, strictly greater than $d$ and strictly less than $d$ (and equal to $d$). We denote the two spatial regions, arbitrarily, left and right.

Similarly, an infinite line embedded in a plane divides this plane in a left and a right planar region (as well as in the linear region defined by the line itself). The same can be said about a point on a line, with respect to this line. The distinction of a left and right side is also applicable to a finite line embedded in a plane, within the proximity of the line, and, similarly, for a finite plane embedded in (a part of) the space $\mathbb{E}^k$. Thus,

**Assertion 3.1** For any given segment or shape, no three (or more) co-equal, disjoint neighborhoods can be found.

**Proof**: Consider $\Delta_1(s)$ and $\Delta_2(s)$ two co-equal, disjoint neighborhoods of a shape $s \in U_{n-1,k}$. Consider the topological space $H$ defined on the $n$-dimensional hyperplane in $\mathbb{E}^k$ that carries both neighborhoods. Under the above isomorphism, the shapes $s$, $\Delta_1(s)$ and $\Delta_2(s)$ define corresponding point sets $S$, $D_1$ and $D_2$, respectively, with $S \subseteq b(D_1)$ and $S \subseteq b(D_2)$, as well as $i(D_1) \cap i(D_2) = \emptyset$.

The Jordan curve theorem (Henle, 1979) specifies that any $n$-dimensional “path” $H_0$ in $H$ divides the hyperplane $H$ into the disjoint subsets $H_0$, $H^-$ and $H^+$. We can choose $H_0$ such that $S \subseteq H_0$, $i(D_1) \subseteq H^-$ and $i(D_2) \subseteq H^+$. Assume there exists a third neighborhood $\Delta_3(s)$ of $s$, co-equal and disjoint to both $\Delta_1(s)$ and $\Delta_2(s)$, with corresponding point set $D_3$.

For every point $p$ in the interior of $S$, there exists a (point set) neighborhood $\delta_n(p)$ such that

\[
\delta_n(p) \cap H_0 \subseteq S, \\
\delta_n(p) \cap H^- \subseteq i(D_1), \\
\delta_n(p) \cap H^+ \subseteq i(D_2).
\]

$\delta_n(p)$ is a subset of $H$, that is,

\[
\delta_n(p) = \delta_n(p) \cap H = \delta_n(p) \cap (H_0 \cup H^- \cup H^+) \\
= (\delta_n(p) \cap H_0) \cup (\delta_n(p) \cap H^-) \cup (\delta_n(p) \cap H^+) \\
\subseteq S \cup i(D_1) \cup i(D_2)
\]
Since \( i(D_3) \cap i(D_1) = \emptyset, i(D_3) \cap i(D_2) = \emptyset, \) and \( i(D_3) \cap S = \emptyset \) it follows that \( i(D_3) \cap \delta_n(p) = \emptyset. \) Thus, \( p \notin b(D_3), \) or, \( b(D_3) \cap S = \emptyset, \) which violates our assumption.  

In the case of a boundary segment, the neighborhoods can be related to the shape of which the segment is a part of the boundary.

**Assertion 3.2** A shape \( s \) in \( U_{n-1} \) is a part of the boundary of a shape \( a \) in \( U_n \) if and only if there exist two neighborhoods of \( s, \) one of which is a part of \( a \) and the other is disjoint of \( a; \)

\[
\forall s \in U_{n-1}, \forall a \in U_n: s \leq B[a] \iff \exists \Delta_1(s), \Delta_2(s) \in U_n: \Delta_1(s) \leq a \land a \cdot \Delta_2(s) = 0.
\]

**Proof:** Consider a shape \( a \) and a part \( s \) of its boundary. Let \( A \) and \( S, \) respectively, denote the corresponding point sets under the isomorphism, with \( S \subseteq b(A). \) Consider a regular point set \( D \) (with rectilinear boundary) that contains \( S \) as well as a neighborhood \( \delta_n(p) \) of a point \( p \in S. \) Since, \( \delta_n(p) \) contains points in \( A \) as well as points in \( A^{-1}, \) so does \( D. \) We have that \( S \) is a subset of the boundary of \( r(D \cap A) \) as well as \( r(D \setminus A). \) Under the isomorphism, \( r(D \cap A) \) and \( r(D \setminus A) \) define (shape) neighborhoods \( \Delta_1(s) \) and \( \Delta_2(s) \) of \( s, \) respectively, with \( \Delta_1(s) \leq a \) and \( \Delta_2(s) \cdot a = 0. \)

Conversely, consider two neighborhoods \( \Delta_1(s) \) and \( \Delta_2(s) \) of \( s, \) with \( \Delta_1(s) \leq a \) and \( \Delta_2(s) \cdot a = 0. \) Let \( D_1 \) and \( D_2, \) respectively, denote the corresponding point sets under the isomorphism, with \( i(D_1) \subseteq i(A) \) and \( i(D_2) \cap i(A) = \emptyset, \) or, \( i(D_2) \subseteq i(A^{-1}). \) Any point \( p \) in \( S \) is near \( i(D_1) \) as well as \( i(D_2), \) and, therefore, near \( A \) and \( A^{-1}. \) It follows that \( p \in b(A), \) or, \( S \subseteq b(A). \) By the isomorphism, \( s \leq B[a]. \)  

Note that the two neighborhoods \( \Delta_1(s) \) and \( \Delta_2(s) \) are necessarily disjoint, for \( \Delta_1(s) \leq a \) and \( a \cdot \Delta_2(s) = 0 \Rightarrow \Delta_1(s) \cdot \Delta_2(s) = 0. \) Also, the shape \( a \) is itself a neighborhood of \( s \) and a part of \( a. \)

From Assertion 3.2, we derive a sufficient condition for a shape to be a part of a boundary.

**Property** Given two shapes \( a \) and \( b \) in \( U_n \) with \( a \) a part of \( b, \) a part \( s \) of the boundary of \( a \) is a part of the boundary of \( b \) if and only if there exists a neighborhood \( \Delta(s) \) of \( s \) such that \( b \cdot \Delta(s) = 0; \)
\( \forall a, b \in U_n, a \leq b, \forall s \in U_{n-1}, s \leq B[a]; s \leq B[b] \iff \exists \Delta(s) \in U_n, b \cdot \Delta(s) = 0. \) [eq. 3.1]

**Proof:** Follows directly from Assertion 3.2.

The third assertion specifies that, when comparing a neighborhood of a shape with another shape co-equal to this neighborhood, it is always possible to partition the neighborhood with respect to the other shape, and to find a corresponding partitioning of the first shape.

---

**Assertion 3.3** Given a shape \( a \in U_n \) a shape \( s \in U_{n-1} \) and a neighborhood \( \Delta(s) \in U_n \) there exists a partitioning of \( s \) into \( t \) and \( s - t \), and a partitioning of \( \Delta(s) \) into neighborhoods \( \Delta(t) \) and \( \Delta(s - t) \), with \( \Delta(t) \leq a \) and \( \Delta(s - t) \cdot a = 0. \)

**Proof:** Consider the shape \( s \) in \( U_{n-1}, k \) and the shapes \( a \) and \( \Delta(s) \) in \( U_{n, k} \). Let \( S, A \) and \( D \), respectively, denote the corresponding point sets in \( E_k \), under the isomorphism. Consider a partitioning of \( D \) with respect to \( A \) into the subsets \( D \cap A \) and \( D \setminus A = D \cap A^{-1} \). We show that \( r(D \cap A) \) and \( r(D \setminus A) \) correspond to the (shape) neighborhoods \( \Delta(t) \) and \( \Delta(s - t) \), respectively, and determine the isomorph point set to \( t \).

Consider any point \( p \) in \( S \). We have that \( p \in b(D) \) and, therefore, \( p \) is near \( i(D) \) and \( i(D^{-1}) \). We can write \( i(D^{-1}) \subseteq i(D^{-1}) \cup i(A^{-1}) \subseteq i(D^{-1} \cup A^{-1}) = i((D \cap A)^{-1}) \) as well as \( i(D^{-1}) \subseteq i(D^{-1}) \cup i(A) \subseteq i(D^{-1} \cup A) = i((D \cap A)^{-1}) = i((D \setminus A)^{-1}) \). It follows that \( p \) is near both \( i((D \cap A)^{-1}) \) and \( i((D \setminus A)^{-1}) \).

If there exists a (point set) neighborhood \( \delta_p(p) \) of \( p \) such that every point in \( \delta_p(p) \cap i(D) \) lies in \( i(A) \), then, \( p \) is near \( i(D) \cap i(A) = i(D \cap A) \). It follows that \( p \in b(i(D \cap A)) \). Consider \( p \notin T, p \in S \setminus T. \)

Otherwise, if there exists a (point set) neighborhood \( \delta_p(p) \) of \( p \) such that every point in \( \delta_p(p) \cap i(D) \) lies in \( i(A^{-1}) \), then, \( p \) is near \( i(D) \cap i(A^{-1}) = i(D \cap A^{-1}) = i(D \setminus A) \). It follows that \( p \in b(i(D \setminus A)) \). Consider \( p \notin T, p \in S \setminus T. \)

If both previous cases fail, then, for every (point set) neighborhood \( \delta_p(p) \) of \( p \), \( \delta_p(p) \cap i(D) \) contains points in \( i(A) \) as well as \( i(A^{-1}) \). Such a point \( p \) belongs to \( b(T) \) as well as \( b(S \setminus T). \)

We have shown that \( r(T) \subseteq b(i(D \cap A)) = b(r(D \cap A)), \) while \( r(S \setminus T) \subseteq b(i(D \setminus A)) = b(r(D \setminus A)). \) It follows from the construction of \( T \) that this point set is isomorphic to a shape \( t \) in \( U_{n-1, k} \), that is a part of \( s. \)
Figure 3.1 illustrates the possible cases that result from Assertion 3.3 for a pair of shapes in $U_2$. Given two shapes $a$ and $b$ with $a$ a part of $b$, and a shape $s$ that is a part of the boundary of $a$, using [eq 3.1], we derive a necessary and sufficient condition for which $s$ is not a part of the boundary of $b$.

**Property**

Given two shapes $a$ and $b$ in $U_n$ with $a$ a part of $b$, for any part $s$ of the boundary of $a$, the product of $s$ and the boundary of $b$ equals 0 if and only if there exists a neighborhood $\Delta(s)$ of $s$, such that $\Delta(s) \leq b - a$;

$$\forall a, b \in U_n, a \leq b, \forall s \in U_{n-1}, s \leq B[a];$$

$$s \cdot B[b] = 0 \iff \exists \Delta(s) \in U_n, \Delta(s) \leq b - a. \quad [eq 3.2]$$

**Proof:** From $s \leq B[a]$ and Assertion 3.2, it follows that $\exists \Delta(s) \in U_n: a \cdot \Delta(s) = 0$. Using Assertion 3.3, we define a partitioning on $s$ and $\Delta(s)$ with respect to $b$ as follows:

$$s = t + (s - t), \Delta(t) = \Delta(s) \cdot b$$

and $\Delta(s - t) = \Delta(s) - b$.

From $\Delta(s - t) \cdot b = 0$, [eq 3.1] specifies that $s - t \leq B[b]$. Since $s \cdot B[b] = 0$, it follows that $t = s$ and $\Delta(t)$ equals a neighborhood $\Delta^*(s)$ of $s$. We have that $\Delta^*(s) \cdot a = 0$, $\Delta^*(s) \leq b$ and, thus, $\Delta^*(s) \leq b - a$.

Conversely, we have that $\exists \Delta(s) \in U_n, \Delta(s) \leq b - a$. Since $s \leq B[a]$, $a$ is a neighborhood of $s$ and $(b - a) \cdot a = 0$. Therefore, it follows from [eq 3.1] that $s \leq B[b - a]$ or $b - a$ is a neighborhood of $s$. Suppose $s \leq B[b]$. Then, $\exists \Delta^*(s) \in U_n: b \cdot \Delta^*(s) = 0$. 

![Figure 3.1](image-url)
We have that \( \Delta^*(s) \cdot (b - a) = 0 \) and, since \( a \leq b \), \( \Delta^*(s) \cdot a = 0 \), which violates Assertion 3.1. Thus, \( s \not\in B[b] \). Since this holds for any \( s \), it follows that \( s \cdot B[b] = 0 \).

For each of the operations of sum, product, difference and symmetric difference, there exists a relation between the boundaries of the operands and the boundary of the result of the operation. Specifically, the boundary of the sum, product, difference or symmetric difference of two shapes is a part of the sum of the boundaries of these shapes.

**Theorem 3.4** The boundary of the shape resulting from one of the operations of sum, product, difference and symmetric difference on two shapes is a part of the sum of the boundaries of these two shapes;

\[
\forall a, b \in U_n: \\
B[a + b] \leq B[a] + B[b] \\
B[a \cdot b] \leq B[a] + B[b] \\
B[a - b] \leq B[a] + B[b] \\
B[a \oplus b] \leq B[a] + B[b]
\]

**Proof:** Take any shape \( s \leq B[a + b] \). We have that \( \exists \Delta(s) \in U_n; \Delta(s) \leq a + b \) and \( \exists \Delta^*(s) \in U_n; \Delta^*(s) \cdot (a + b) = 0 \). It follows that \( \Delta^*(s) \cdot a + \Delta^*(s) \cdot b = 0 \), or \( \Delta^*(s) \cdot a = 0 \land \Delta^*(s) \cdot b = 0 \). Using Assertion 3.3, we define a partitioning on \( s \) and \( \Delta(s) \) with respect to \( a \) as follows:

\[
s = t + (s - t), \quad \Delta(t) = \Delta(s) \cdot a \text{ and } \Delta(s - t) = \Delta(s) - a.
\]

Also, we have that \( \Delta(t) \leq a \) while \( \Delta^*(t) \cdot a = 0 \) (\( \Delta^*(t) \leq \Delta^*(s) \)). It follows from Assertion 3.2 that \( t \) is a part of the boundary of \( a \).

Given that \( \Delta(s - t) \leq \Delta(s) \leq (a + b) \) and \( \Delta(s - t) \cdot a = 0 \), it follows that \( \Delta(s - t) \leq b \) while \( \Delta^*(s - t) \cdot b = 0 \) (\( \Delta^*(s - t) \leq \Delta^*(s) \)). It follows from Assertion 3.2 that \( s - t \) is a part of the boundary of \( b \). Thus \( s \leq B[a] + B[b] \), or \( B[a + b] \leq B[a] + B[b] \).

Take any shape \( s \leq B[a \cdot b] \). We have that \( \exists \Delta(s) \in U_n; \Delta(s) \cdot (a \cdot b) = 0 \). Using Assertion 3.3, we define a partitioning on \( s \) and \( \Delta(s) \) with respect to \( a \) as follows:

\[
s = t + (s - t), \quad \Delta(t) = \Delta(s) \cdot a \text{ and } \Delta(s - t) = \Delta(s) - a.
\]

We have that \( \Delta(t) \cdot b = (\Delta(s) \cdot a) \cdot b = 0 \), while \( t \leq s \leq B[a \cdot b] \) and \( a \cdot b \leq b \). It follows from [eq3.1] that \( t \) is a part of the boundary of \( b \).
Since \( \Delta(s - t) \cdot a = 0 \) and \( a \cdot b \leq a \), while \( s - t \leq s \leq B[a \cdot b] \), [eq3.1] specifies that \( s - t \) is a part of the boundary of \( a \). Thus \( s \leq B[a] + B[b] \), or \( B[a \cdot b] \leq B[a] + B[b] \).

Take any shape \( s \leq B[a - b] \). We have that \( \exists \Delta(s) \in U_n : \Delta(s) \cdot (a - b) = 0 \). Using Assertion 3.3, we define a partitioning on \( s \) and \( \Delta(s) \) with respect to \( b \) as follows:

\[
s = t + (s - t), \quad \Delta(t) = \Delta(s) \cdot b \quad \text{and} \quad \Delta(s - t) = \Delta(s) - b.
\]

Since \( t \leq B[a - b] \), \( a - b \) is a neighborhood of \( t \) and we have that \( (a - b) \cdot b = 0 \). Given that \( \Delta(t) \leq b \), [eq3.1] specifies that \( t \) is a part of the boundary of \( b \).

We have that \( \Delta(s - t) \cdot b = 0 \) as well as \( \Delta(s - t) \cdot (a - b) = 0 \). Thus, \( \Delta(s - t) \cdot a = 0 \) and \( a \cdot b \leq a \), while \( s - t \leq s \leq B[a - b] \). It follows from [eq3.1] that \( s - t \) is a part of the boundary of \( a \). Thus \( s \leq B[a] + B[b] \), or \( B[a - b] \leq B[a] + B[b] \).

\[
B[a \oplus b] = B[(a - b) + (b - a)] \leq B[a - b] + B[b - a] \leq B[a] + B[b].
\]

Figure 3.2 illustrates these results for two shapes in \( U_2 \). Given a measure of the boundary of a shape, we conclude that the “size” of the boundary of a shape that is the result of an operation from among sum, product, difference and symmetric difference, is bounded from above by the sum of the “sizes” of the boundaries of the operand shapes.
3.1.3 Segment

In the sequel we define a segment as a shape with a “minimal” boundary, define a representation of a shape as a set of disjoint segments and relate the boundary of a shape to the boundaries of the segments in its representation.

We define a boundary shape to denote any shape in $U_n$ that is the boundary of a shape in $U_{n+1}$. We use the term boundary to denote both a mapping of a shape in $U_n$ to a shape in $U_{n-1}$ (serving as a part of the representation of the shape in $U_n$) and to denote a boundary shape in $U_n$. In the latter case we use the term boundary most often in conjunction with one of the adjectives “simple”, “inner” or “outer” to specify a special instance of a boundary shape.

**Definition 3.4** A boundary shape\(^8\) or, simply, boundary is a shape $a$ in $U_n$ for which there exists a shape $b$ in $U_{n+1}$ such that $a = B[b]$; we say that the boundary $a$ defines the shape $b$ and write $b = \Gamma(a)\(^9\).

**Property** The empty boundary shape defines only the empty shape; $\Gamma(0) = 0$.

Then, a segment is defined as a shape with a “minimal” boundary with respect to this shape. That is, no part of the boundary can be found that defines a part of the shape or segment. A segment is necessarily co-equal.

**Definition 3.5** A shape $a$ is a segment if and only if there exists no shape $b, b \neq 0$ and $b \neq a$, such that $b$ is a part of $a$ and the boundary of $b$ is a part of the boundary of $a$ ($b \leq a \land B[b] \leq B[a]$).

Figure 3.3 illustrates this definition with a few shapes in $U_2$. For shapes in $U_1$, a minimal boundary shape of a segment consists of two points, denoted the *endpoints* of the line segment. Their isomorph points in $E^k$ both lie on the line segment’s carrier. Given a segment $l$, let $tail[l]$ and $head[l]$ denote the two endpoints with $tail[l] \leq_c head[l]$.

---

8. Even though we use the term boundary segment to denote a segment that is a part of the boundary of a shape, we avoid using the term boundary shape to denote only a part of a shape’s boundary.

9. $\Gamma$ may be viewed as a “constructor” function.
3.1 Shape Representation

Definition 3.6 A line segment \( l \) is embedded in a line segment \( m \) if and only if \( l \) and \( m \) are co-equal, \( \text{tail}(m) \leq_c \text{tail}(l) \) and \( \text{head}(l) \leq_c \text{head}(m) \).

Two co-equal line segments \( l_1 \) and \( l_2 \) are disjoint if either \( \text{head}(l_1) \leq_c \text{tail}(l_2) \) or \( \text{head}(l_2) \leq_c \text{tail}(l_1) \). If two line segments intersect, they necessarily have different co-descriptors and, therefore, are disjoint. Figure 3.4 illustrates the concepts of embedding and disjointedness for line segments.

Lemma 3.5 For any shape \( a \) there exists a finite set \( \{a_1, \ldots, a_m\} \) of disjoint segments such that \( a \) equals the sum of \( a_1, \ldots, a_m \):

\[
\forall a \in U_n \exists \text{segments } a_1, \ldots, a_m \in U_n: a_1 \cdot a_j = 0 \forall i \neq j \land a = a_1 + \ldots + a_m
\]

Proof: If \( a \) is a segment, then \( a = a \) proves the proposition.

Otherwise, there exists a shape \( b \neq 0 \) with \( b \neq a, b \leq a \) and \( B[b] \leq B[a] \). Consider the shapes \( b \) and \( a - b \) and assume both can be written as a sum of disjoint segments: \( b = b_1 + \ldots + b_m \) and \( a - b = c_1 + \ldots + c_m \). Since \( b \cdot (a - b) = 0 \), it follows that \( b_i \cdot c_j = 0 \) for all \( i \leq m' \) and...
Thus, \( a = b + (a - b) = b_1 + \ldots + b_{m'} + c_1 + \ldots + c_{m''} = a_1 + \ldots + a_m \) with \( m = m' + m'' \) and all \( a_i \) (\( b_i \) and \( c_i \)) are disjoint.

In the same way, we prove that both \( b \) and \( a - b \) can be written as a sum of disjoint segments. This induction is finite as both \( b \) and \( a - b \) are a part of \( a \), neither are equal to \( a \) and the boundary of \( a \) is bounded (a boundary is a shape and any shape is bounded).

We say that the set \( \{a_1, \ldots, a_m\} \) represents the shape \( a \) and we write \( a \{a_1, \ldots, a_m\} \) to denote the shape \( a \) with representation \( \{a_1, \ldots, a_m\} \). As such, \( a \{a_1, \ldots, a_m\} \) constitutes two alternative views of the same object or shape, one algebraic and one representational, respectively. Note that, since \( a_1, \ldots, a_m \) are disjoint, we can write \( a = a_1 \oplus \ldots \oplus a_m \).

**Definition 3.7** The set \( \{a_1, \ldots, a_m\} \) of segments \( a_i \in U_n \) (\( i \leq m \)) represents a shape \( a \in U_n \) if and only if \( a = a_1 + \ldots + a_m \) and \( a_i \cdot a_j = 0 \) \( \forall i, j \) with \( i \neq j \). Then, \( \{a_1, \ldots, a_m\} \) is denoted a representation for \( a \).

Given a co-equal shape and its representation, the boundary of the shape can be derived from the boundaries of the segments in the representation using the operation of symmetric difference. Figure 3.5 illustrates the representation of a shape as a set of segments and the relation of the shape’s boundary with the boundaries of the segments.

**Lemma 3.6** The boundary of a co-equal shape \( a \{a_1, \ldots, a_m\} \) equals the symmetric difference of the boundaries \( B[a_i] \) (\( i \leq m \));

\[
\forall a \{a_1, \ldots, a_m\} \in U_n, co[a_i] = co[a_j] \forall i, j \leq m : B[a] = B[a_1] \oplus \ldots \oplus B[a_m].
\]

**Proof:** If \( m = 1 \), then \( B[a] = B[a_1] \).
If \(m = 2\), we prove that \(B[a] = B[a_1] \oplus B[a_2]\), or, \(B[a] = (B[a_1] + B[a_2]) - (B[a_1] \cdot B[a_2]) = (B[a_1] - B[a_2]) + (B[a_2] - B[a_1])\).

First, consider a segment \(s \leq B[a_1] - B[a_2]\). Since \(s \leq B[a_1]\), there exist two neighborhoods \(\Delta_1(s)\) and \(\Delta_2(s)\) with \(\Delta_1(s) \leq a_1 = a_1 + a_2\) and \(\Delta_2(s) \cdot a_1 = 0\) (Assertion 3.2). We partition \(\Delta_2(s)\) with respect to \(a_2\) into the neighborhoods \(\Delta_2(t) \leq a_2\) and \(\Delta_2(s - t)\), with \(\Delta_2(s - t) \cdot a_2 = 0\) and \(t \leq s\) (Assertion 3.3). Since \(\Delta_2(s - t) \leq \Delta_2(s)\), it follows that \(\Delta_2(s - t) \cdot a_1 = 0\) and, thus, \(\Delta_2(s - t) \cdot a = 0\). Also, since \(\Delta_2(t) \leq \Delta_2(s)\), we have that \(\Delta_2(t) \cdot a_1 = 0\). Given that \(t \cdot B[a_2] = 0\) and \(\Delta_2(t) \leq a_2\), consider a neighborhood \(\Delta(t)\) with \(\Delta(t) \cdot t = 0\) and \(\Delta(t) \leq a_2\). Then, Assertion 3.1 specifies that \(\Delta(t) \cdot a_1 \neq 0\). However, this implies that \(a_1 \cdot a_2 \neq 0\), which contradicts our assumption. Therefore, \(t = 0\) and \(\Delta_2(s - t)\) is a neighborhood of \(s\), or, \(\Delta_2(s - t) = \Delta(s)\). Given \(\Delta_1(s) \leq a\) and \(\Delta(s) \cdot a = 0\), \((\Delta_1(s) \cdot \Delta(s) = 0)\), it follows from Assertion 3.2 that \(s \leq a\).

The proof is similar for a segment \(s \leq B[a_2] - B[a_1]\).

Finally, consider a segment \(s \leq B[a_1] \cdot B[a_2]\). Then, we can construct two neighborhoods \(\Delta_1(s)\) and \(\Delta_2(s)\) with \(\Delta_1(s) \cdot \Delta_2(s) = 0\) such that \(\Delta_1(s) \leq a_1\) and \(\Delta_2(s) \leq a_2\). (Otherwise, consider \(\Delta_1(s) \leq a_1\) and \(\Delta_2(s) \cdot a_1 = 0\); partition \(\Delta_2(s)\) with respect to \(a_2\) into \(\Delta_2(t) \leq a_2\) and \(\Delta_2(s - t)\), with \(\Delta_2(s - t) \cdot a_2 = 0\) and \(t \leq s\) (Assertion 3.3). However, since \(\Delta_1(s) \leq a_1\) and \(a_1 \cdot a_2 = 0\), it follows that \(\Delta_1(s) \cdot a_2 = 0\). Consider \(\Delta_1(s - t) \leq \Delta_1(s)\); then, \(\Delta_1(s - t) \cdot a_2 = 0\) as well as \(\Delta_2(s - t) \cdot a_2 = 0\) and \(\Delta_2(s - t) \cdot a_2 = 0\). Thus, from Assertion 3.1 and Assertion 3.2, \(s \neq B[a_2]\), which contradicts \(s \leq B[a_1] \cdot B[a_2]\). It follows that both \(\Delta_1(s) \leq a_1 \leq a\) and \(\Delta_2(s) \leq a_2 \leq a\) and, thus, \(s \leq B[a_1] = 0\) (Assertion 3.1 and Assertion 3.2, negated).

If \(m > 2\), let \(a'\) denote the shape \(a_1 + \ldots + a_{m-1}\) and assume \(B[a'] = B[a_1] \oplus \ldots \oplus B[a_{m-1}]\). Then, \(a = a' + a_m\) and, given the previous result, \(B[a] = B[a'] \oplus B[a_m] = B[a_1] \oplus \ldots \oplus B[a_m]\) is proven by induction.

### 3.2 Maximal Segment Representation

The maximal representation is a canonical representation of a shape as a set of (disjoint) segments, each represented by its co-descriptor and boundary (Krishnamurti, 1992a). Each segment’s boundary is again a shape (or composed of a set of shapes, each) subject to the
maximal representation (in $U_{n-1}$). This leads to a recursive set of definitions, for which points or segments in $U_0$ serve as a base case.

### 3.2.1 Canonical Representation

A segment is maximal if it cannot be combined under the operation of sum with any other segment in the representation to form a single segment. The following lemma states the conditions under which two co-equal segments combine to form a single segment, under the operation of sum. Figure 3.6 illustrates these conditions for shapes in $U_3$. Segments that are not co-equal cannot combine to form a single segment.

**Lemma 3.7** The sum of two co-equal segments is a segment if and only if the segments are not disjoint or their boundaries are not disjoint.

**Proof:** Consider the sum $a + b$ of two co-equal segments $a$ and $b$. If either $B[a] \leq B[a + b]$ or $B[b] \leq B[a + b]$, then $a + b$ is not a segment.

**Figure 3.6** The sum of two segments is a segment if the segments are not disjoint (a, d) or their boundaries are not disjoint (b) and is not a segment, otherwise (c, e, f).
First we prove it is a necessary condition: If $a$ and $b$ are disjoint, the boundary of $a + b$ equals the symmetric difference of the boundaries of $a$ and $b$ (from Lemma 3.6), $B[a + b] = B[a] \oplus B[b] = (B[a] + B[b]) - (B[a] \cdot B[b])$. If the boundaries $B[a]$ and $B[b]$ are also disjoint, then, we have that $B[a + b] = B[a] \oplus B[b] = B[a] + B[b]$. It follows that $B[a] \leq B[a + b]$ (and $B[b] \leq B[a + b]$) and, therefore, $a + b$ is not a segment (by definition).

Assume $a$ and $b$ are disjoint, i.e. $a \cdot b = 0$, but their boundaries are not disjoint, i.e., $B[a] \cdot B[b] \neq 0$. We have that $B[a + b] = B[a] \oplus B[b] = B[a] + B[b] - (B[a] \cdot B[b])$ and, therefore, $B[a] \not\subseteq B[a + b]$ and $B[b] \not\subseteq B[a + b]$. Assume there exists a shape $d 
eq 0$ with $d \not= a + b$, $d \leq a + b$ and $B[d] \leq B[a + b] + B[a + b]$. Consider $f = d \cdot a$ and $g = d \cdot b$, then $f \cdot g = 0$ and $d = f + g$. It follows that $B[f] \leq B[a] + B[d] \leq B[a] + B[b]$ and idem for $B[g]$. Consider a segment $s \leq B[f] \cdot B[b]$. Since $f \cdot b = 0$, there exist two neighborhoods $\Delta_1(s)$ and $\Delta_2(s)$ with $\Delta_1(s) \cdot \Delta_2(s) = 0$, $\Delta_1(s) \leq f$ and $\Delta_2(s) \leq b$. It follows that $\Delta_1(s) \leq a$ and $\Delta_2(s) \cdot a = 0$, or $s \leq B[a]$. We conclude that $B[f] \leq B[a]$, which contradicts the definition of a segment (given $f \leq a$), and similarly for $g$. Thus, no such shape $d$ exists and, by definition, $a + b$ is a segment.

Otherwise, $a$ and $b$ are not disjoint and there exists a set $\{c_1, \ldots, c_m\}$ of disjoint segments such that $a + b$ equals the sum of $c_1, \ldots, c_m$. Let $c_i$ and $c_j$, with $i < j$, denote any two segments such that $B[c_i] \cdot B[c_j] \neq 0$. Using the previous result, it follows that $c_i + c_j$ is a segment. Thus, $\{c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_{j-1}, c_{j+1}, \ldots, c_m, c_i + c_j\}$ is a set of disjoint segments that represents $a + b$. Repeatedly, we replace, in the representation of $a + b$, any two segments $x$ and $y$, with $B[x] \cdot B[y] \neq 0$, with the sum $x + y$. This results in a representation of $a + b$ by a set $\{d_1, \ldots, d_m\}$ of disjoint segments, with $B[d_i] \cdot B[d_j] = 0$ for all $i \neq j$.

We have that $a = d_1 \cdot a + \ldots + d_m \cdot a$. Assume that $B[d_i \cdot a] \cdot B[d_j \cdot a] = s \neq 0$. Then, there exist two neighborhoods $\Delta_1(s)$ and $\Delta_2(s)$ with $\Delta_1(s) \cdot \Delta_2(s) = 0$, $\Delta_1(s) \leq d_i \cdot a \leq d_i$ and $\Delta_2(s) \leq d_j \cdot a \leq d_j$. Since $d_i \cdot d_j = 0$, we have that $s \leq B[d_i] \cdot B[d_j]$, which contradicts the fact that $B[d_i] \cdot B[d_j] = 0$ for all $i \neq j$. Thus, $B[d_i \cdot a] \cdot B[d_j \cdot a] = 0$ for all $i \neq j$. However, this contradicts the fact that $a$ is a segment. Therefore, the representation of $a + b$ must consist of a single segment, which is equal to $a + b$. \\
Theorem 3.8 continues from Lemma 3.5 and describes a shape as the sum of disjoint segments with disjoint boundaries. It serves as the basis for the maximal representation of shapes (see Definition 3.13 in Section 3.2.3).

**Theorem 3.8** For any shape a there exists exactly one\(^{10}\) finite set \{a\(_1\), ..., a\(_m\)\} of disjoint segments with disjoint boundaries such that a equals the sum of a\(_1\), ..., a\(_m\):

\[
\forall a \in U_m \exists \{a\(_1\), ..., a\(_m\)\}, \text{ segments } a\(_1\), ..., a\(_m\) \in U_m:

(a\(_1\) · a\(_j\) = 0 ∧ B[a\(_1\)] · B[a\(_j\)] = 0) ∨ i ≠ j ∧ a = a\(_1\) + ... + a\(_m\).
\]

**Proof:** First we prove that such a set exists: Given a shape a we can find a set \{a\(_1\), ..., a\(_m\)\} of disjoint segments such that \(a = a_1 + ... + a_m\) (Lemma 3.5). If \(B[a_i] · B[a_j] = 0\) for all \(i ≠ j\), then, the above is proven. Otherwise, take any two segments \(a_i\) and \(a_j\) (\(i < j\)) with \(B[a_i] · B[a_j] ≠ 0\). Lemma 3.7 specifies that \(b = a_i + a_j\) is a segment.

Thus, \{a\(_1\), ..., a\(_{i-1}\), a\(_{i+1}\), ..., a\(_{j-1}\), a\(_{j+1}\), ..., a\(_m\), b\} is a set of disjoint segments with \(a = a_1 + ... + a_{i-1} + a_{i+1} + ... + a_{j-1} + a_{j+1} + ... + a_m + b\). By induction, this results in a set \{b\(_1\), ..., b\(_m\)\} of disjoint segments with disjoint boundaries such that \(a = b_1 + ... + b_m\).

Secondly, assume there exist two sets \{a\(_1\), ..., a\(_m\)\} and \{b\(_1\), ..., b\(_m\)\}. Then, at least one segment \(b_1\) is different from all segments \(a_j\) and we write \(b_1 = b_1 · a_1 + ... + b_1 · a_m = c_1 + ... + c_m^r\) (with \(c_1\) equal to the first \(b_1 · a_j\) that is different from 0, and so on.). \(c_1, ..., c_m^r\) are disjoint segments with not all disjoint boundaries (otherwise, \(b_1\) is not a segment).

Assume \(B[c_k] · B[c_j] ≠ 0\). Consider a segment \(s ≤ B[c_k] · B[c_j]\). Then, given that \(c_k · c_j = 0\), there exist two neighborhoods \(Δ_1(s)\) and \(Δ_2(s)\) with \(Δ_1(s) · Δ_2(s) = 0, Δ_1(s) ≤ c_k\) and \(Δ_2(s) · c_j = 0\) for \(Δ_1(s) ≤ c_i\), and \(Δ_2(s) ≤ c_k\). Assume \(c_k = b_1 · a_k^r\) and \(c_j = b_1 · a_k\).

Since \(Δ_1(s) ≤ c_k \leq b_k\), we have that \(Δ_1(s) · c_j = Δ_1(s) · b_1 · a_k = Δ_1(s) · a_k = 0\) as well as \(Δ_2(s) · c_k ≤ Δ_2(s) · c_j\), and similarly for \(a_k^r\). Thus, \(s ≤ B[a_k] · B[a_k^r]\), which contradicts our assumption.

Above, we have defined a segment using the notion of a minimal boundary with respect to a shape and its boundary. We can also give a semi-constructive definition of a segment using the notion of a *simple boundary*, that is, a “minimal” boundary shape as a part of a boundary.

---

10. The two-character symbol “∃!” denotes “there exists exactly one.”
shape, not with respect to the defined shape. In Section 7.1 (Algorithm 7.1) and Section 7.2 (Algorithm 7.4), we describe the construction of the simple boundaries for a plane segment and volume segment, respectively.

**Definition 3.8** A boundary shape \( a \) is a simple boundary if and only if there exists no boundary shape \( b, b \neq 0 \) and \( b \neq a \), that is a part of \( a \).

**Property** A simple boundary \( a \) in \( U_n \) defines a segment \( \Gamma(a) \) in \( U_{n+1} \).

*Proof:* Follows directly from the definitions of segment and simple boundary. ❏

However, the boundary shape of a segment is not always a simple boundary. Figure 3.7 illustrates this with a few segments in \( U_2 \). Yet, the boundary of a co-equal shape can always be described as composed of a finite set of disjoint simple boundaries.

**Lemma 3.9** For any boundary shape \( b \) there exists a set \( \{b_1, ..., b_m\} \) of disjoint, simple boundaries such that \( b \) is the composition of \( b_1, ..., b_m \) under the operation of sum or symmetric difference.

\[
\forall \text{ boundary } b \in U_n \exists \text{ simple boundaries } b_1, ..., b_m \in U_n: \\
b_i \cdot b_j = 0 \forall i \neq j \land b = b_1 + ... + b_m
\]

*Proof:* We prove that the boundary of any segment can be decomposed into disjoint, simple boundaries. Then, from Theorem 3.8, it follows that the boundary of any shape, i.e., any boundary shape, can be decomposed into disjoint, simple boundaries. Let \( a \) denote the boundary of a segment \( \Gamma(a) \). If \( a \) is a simple boundary, then the proposition is proven.
Otherwise, there exists a boundary shape $b \neq 0$ with $b \neq a$ and $b \leq a$. Since $B[\Gamma(b) \cdot \Gamma(a)] \leq b + a \leq a$ and $\Gamma(a)$ is a segment, it follows (from Definition 3.5) that either $\Gamma(b) \cdot \Gamma(a) = 0$ or $\Gamma(b) \cdot (\Gamma(a) = \Gamma(a))$. If $\Gamma(b) \cdot \Gamma(a) = 0$, we have that $\Gamma(b) + \Gamma(a) = \Gamma(b) \oplus \Gamma(a)$ and $B[\Gamma(b) + \Gamma(a)] = b \oplus a = a - b (b \leq a)$. Thus, $a - b$ is a boundary shape and, by induction, the proposition is proven. Otherwise, $\Gamma(b) \cdot \Gamma(a) = \Gamma(a)$ and we prove that $a \cdot b = B[\Gamma(b) \cdot \Gamma(a)]$.

Consider a segment $s \leq a - b$. It follows that $s \leq a$ and $s \cdot b = 0$. Then, since $\Gamma(a) \leq \Gamma(b)$, there exist two neighborhoods $\Delta_1(s)$ and $\Delta_2(s)$ with $\Delta_1(s) \cdot \Delta_2(s) = 0$, $\Delta_1(s) \leq \Gamma(a) \leq \Gamma(b)$, $\Delta_2(s) \cdot \Gamma(a) = 0$ and $\Delta_2(s) \leq \Gamma(b)$. Therefore, $\Delta_1(s) \cdot (\Gamma(b) - \Gamma(a)) = 0$ while $\Delta_2(s) \leq \Gamma(b)$ and, thus $s \leq B[\Gamma(b) - \Gamma(a)]$, or, $a - b \leq B[\Gamma(b) - \Gamma(a)]$.

Theorem 3.4 specifies that $B[\Gamma(b) - \Gamma(a)] \leq a + b \leq a$. Consider a segment $s \leq B[\Gamma(b) - \Gamma(a)]$. It follows that $s \leq a$. Then, since $\Gamma(a) \leq \Gamma(b)$, there exist two neighborhoods $\Delta_1(s)$ and $\Delta_2(s)$ with $\Delta_1(s) \cdot \Delta_2(s) = 0$ and $\Delta_1(s) \leq \Gamma(b) - \Gamma(a)$, i.e., $\Delta_1(s) \leq \Gamma(b)$ and $\Delta_2(s) \cdot \Gamma(a) = 0$, $\Delta_2(s) \cdot (\Gamma(b) - \Gamma(a)) = 0$ and $\Delta_2(s) \leq \Gamma(a)$. Therefore, $\Delta_1(s) \leq \Gamma(b)$ and $\Delta_2(s) \leq \Gamma(a) \leq \Gamma(b)$ and, thus, $s \cdot b = 0$, or, $s \leq a - b$, such that $B[\Gamma(b) - \Gamma(a)] \leq a - b$.

We conclude that $B[\Gamma(b) - \Gamma(a)] = a - b$, or, $a - b$ is a boundary shape and, by induction, the proposition is proven.

Consider a simple boundary $b$ that is a part of a boundary $a$ defining a segment $\Gamma(a)$. Using the results from the previous proof, we conclude that either $\Gamma(a) \cdot \Gamma(b) = 0$ or $\Gamma(a) \leq \Gamma(b)$. If $\Gamma(a) \leq \Gamma(b)$, we say that $b$ is an outer boundary; otherwise, $b$ is an inner boundary. Figure 3.8 illustrates the classification into inner and outer boundaries of a set of non-intersecting, simple boundaries in $U_1$.

**Definition 3.9** A simple boundary $b$ that is a part of the boundary of a segment $a$ is an outer boundary for $a$ if $a \leq \Gamma(b)$ and is an inner boundary for $a$, otherwise.

A segment has a single outer boundary and either zero, one or more inner boundaries. Figure 3.9 illustrates this with a few segments and shapes in $U_2$. 
Lemma 3.10

Given a segment \( a \) there exists exactly one simple boundary that is an outer boundary for \( a \).

\[ \forall \text{ segment } a \in U_m \exists ! \text{ simple boundary } b \in U_{m-1}: b \leq B[a] \wedge a \leq \Gamma(b) \]

Proof: First, we prove there exists at least one outer boundary for \( a \).

There exists a set \( \{ b_1, ..., b_m \} \) of disjoint, simple boundaries such that \( B[a] = b_1 + ... + b_m \).

If none of \( b_1, ..., b_m \) is an outer boundary, then \( \Gamma(b_i) \cdot a = 0 \) for all \( i \leq m \). Consider \( a' = a + \Gamma(b_1) + ... + \Gamma(b_m) \). We have that \( B[a'] \leq B[a] = b_1 + ... + b_m \). Consider a segment...
s ≤ B[a'] ≤ B[a]. There exist two neighborhoods ∆₁(s) and ∆₂(s) (∆₁(s) · ∆₂(s) = 0) with Δ₁(s) · a' = 0 and, therefore, Δ₁(s) · a = 0, and Δ₂(s) ≤ a ≤ a'. It follows that both Δ₁(s) · b_i = 0 (from Δ₁(s) · a' = 0) and Δ₂(s) · b_j = 0 (from Δ₂(s) ≤ a) for all i ≤ m. Thus, s ∉ b_i and s ∉ B[a] = b_1 + ... + b_m. Given that a' ≠ 0, it must be that either B[a] ≠ b_1 + ... + b_m or not all b_1, ..., b_m are inner boundaries.

Assume there exist two outer boundaries b and b' for a. We have that a ≤ Γ(b) and a ≤ Γ(b'). Consider Γ(b) ∉ Γ(b'), i.e., Γ(b') − Γ(b) ≠ 0. Since b is a simple boundary, it follows that B[Γ(b') − Γ(b)] ≠ b. Also, since Γ(b') − Γ(b) ≠ Γ(b), we have that B[Γ(b') − Γ(b)] ≠ b. Thus, there exists a boundary segment s ≤ B[Γ(b') − Γ(b)] − b and a neighborhood Δ₁(s) with Δ₁(s) ≤ Γ(b') − Γ(b). Since Δ₁(s) · Γ(b) = 0 and s · b = 0, a neighborhood Δ₂(s) exists with Δ₁(s) · Δ₂(s) = 0 and Δ₂(s) · Γ(b) = 0. It follows that both Δ₁(s) · a = 0 and Δ₂(s) · a = 0, or, s ∉ a, which contradicts our assumption. The proof is similar for Γ(b') ∉ Γ(b).

The canonical representation of a shape consists of a unique representation of the shape as a set of segments as well as a unique representation of each segment as a set of simple boundaries.

**Theorem 3.11** For any segment a in U_n there exists exactly one finite set {b_1, ..., b_m} of disjoint, simple boundaries in U_{n-1} such that B[a] equals the sum of b_1, ..., b_m:

∀ segment a ∈ U_n ∃ {b_1, ..., b_m}, simple boundaries b_1, ..., b_m ∈ U_{n-1}:

\[ b_1 · b_j = 0 \forall i ≠ j \land B[a] = b_1 + ... + b_m. \]

**Proof:** Lemma 3.10 specifies that exactly one outer boundary exists for a. We denote this outer boundary b_0. Consider the shape b = B[a] − b_0. There exists a finite set {b_1, ..., b_{m-1}} of disjoint, simple boundaries such that B[a] = b_0 + b = b_0 + b_1 + ... + b_{m-1} (Lemma 3.9). Consider the set \{Γ(b_1), ..., Γ(b_{m-1})\} of segments defined by b_1, ..., b_{m-1}. If these segments are all disjoint, then, it follows from Theorem 3.8 that this decomposition into simple boundaries is unique.

Assume there exist two segments Γ(b_i) and Γ(b_j) (i ≠ j) that are not disjoint, i.e., Γ(b_i) · Γ(b_j) ≠ 0. Then, a segment s ≤ B[Γ(b_i) · Γ(b_j)] ≤ b_i + b_j ≤ a and two neighborhoods Δ₁(s) and Δ₂(s) exist with Δ₁(s) · Δ₂(s) = 0, Δ₁(s) ≤ Γ(b_i) · Γ(b_j) and Δ₂(s) · Γ(b_i) · Γ(b_j) = 0.
Using Assertion 3.3, we partition $\Delta_2(s)$ with respect to $\Gamma(b_i) + \Gamma(b_j)$ into the neighborhoods $\Delta_2(t)$ and $\Delta_2(s - t)$ with respect to $\Gamma(b_i) + \Gamma(b_j)$ and $\Delta_2(s - t) \cdot (\Gamma(b_i) + \Gamma(b_j)) = 0$. From [eq2.24], it follows that $(\Delta_2(s - t) \cdot \Gamma(b_i)) + (\Delta_2(s - t) \cdot \Gamma(b_j)) = 0$, or, $\Delta_2(s - t) \cdot \Gamma(b_i) = 0$ and $\Delta_2(s - t) \cdot \Gamma(b_j) = 0$. However, this implies that both $s - t \leq b_i$ and $s - t \leq b_j$ which contradicts the hypothesis of disjoint boundaries. Thus, we have that $\Delta_2(t) = \Delta(s) \leq \Gamma(b_i) + \Gamma(b_j)$, while $\Delta_1(s) \leq \Gamma(b_i) \cdot \Gamma(b_j)$ and $\Delta_1(s) \cdot \Delta(s) = 0$. Since $b_i$ and $b_j$ are inner boundaries, i.e., $a \cdot \Gamma(b_i) = 0$ and $a \cdot \Gamma(b_j) = 0$, it follows that both $\Delta_1(s) \cdot a = 0$ and $\Delta(s) \cdot a = 0$, such that $s \not\leq a$. This contradicts the assumption $s \leq b_i + b_j \leq a$ and, thus, $\Gamma(b_i)$ and $\Gamma(b_j)$ must be disjoint. 

Note that this only holds for segments. Figure 3.10 illustrates a shape in $U_2$, the boundary of which can be represented as two different sets of disjoint, simple boundaries.

The set of simple boundaries representing a segment’s boundary defines a set of segments that relate in a unique way to the original segment.

**Property** Given a segment $a$ in $U_n$ and a finite set $\{b_0, b_1, \ldots, b_{m-1}\}$ of simple boundaries in $U_{n-1}$, with $B[a] = b_0 + b_1 + \ldots + b_{m-1}$ and $b_0$ the outer boundary for $a$, it holds that $a = \Gamma(b_0) - (\Gamma(b_1) + \ldots + \Gamma(b_{m-1}))$.

**Proof:** Given $b_0$ the outer boundary for $a$ and $b_1, \ldots, b_{m-1}$ all inner boundaries for $a$, we have that $a \leq \Gamma(b_0)$ and $a \cdot \Gamma(b_i) = 0$ for all $1 \leq i \leq m-1$, or, $a \leq \Gamma(b_0) - (\Gamma(b_1) + \ldots + \Gamma(b_{m-1}))$.

Consider the shape $c = \Gamma(b_0) - (\Gamma(b_1) + \ldots + \Gamma(b_{m-1})) - a$ with $c \cdot a = 0$. If $c \neq 0$, then, there exists a segment $s \leq B[c] \leq B[a]$ and two neighborhoods $\Delta_1(s)$ and $\Delta_2(s)$ ($\Delta_1(s) \cdot \Delta_2(s) = 0$)
with $\Delta_1(s) \leq c$, $\Delta_1(s) \cdot a = 0$, $\Delta_2(s) \leq a$ and $\Delta_2(s) \cdot c = 0$. Given that $a \leq \Gamma(b_0) - (\Gamma(b_1) + \ldots + \Gamma(b_{m-1}))$, we have that $\Delta_2(s) \leq a \leq \Gamma(b_0)$ as well as $\Delta_1(s) \leq c \leq \Gamma(b_0)$, such that $s \not\in b_0$. Similarly, for all $1 \leq i \leq m-1$, it holds that $\Delta_1(s) \cdot \Gamma(b_0) = 0$ as well as $\Delta_2(s) \cdot \Gamma(b_0) = 0$, such that $s \not\in b_i$. Since $s \not\in b_i$ for all $0 \leq i \leq m-1$ and given that $B[a] = b_0 + b_1 + \ldots + b_{m-1}$, it follows that $s \not\in B[a]$, which contradicts the assumption. Thus, $c = 0$ and $a = \Gamma(b_0) - (\Gamma(b_1) + \ldots + \Gamma(b_{m-1}))$. 

$\square$

### 3.2.2 Geometric Relations

Lemma 3.7 specifies that two co-equal segments can be combined into a single segment only if they are not disjoint or their boundaries are not disjoint. We say that two disjoint segments share boundary if their boundaries are not disjoint. At the same time, we redefine the notion of disjointedness and say that two segments are disjoint only if they do not share boundary (nor overlap or one contains the other). In this section, we give the proper definitions for share boundary and disjoint, as we use them from here on. Next, in Section 3.2.3, we define the maximal representation and give some properties of the representation.

In Section 2.2, we have defined the (algebraic) shape relations contain, overlap and disjoint (on shapes within the same algebra): A shape contains another shape if the second shape is a part of the first shape and two shapes overlap if their product is non-zero and one shape does not contain the other. Otherwise, two shapes are disjoint.

Geometrically, we can relate two shapes as to whether their boundaries overlap or not.

**Definition 3.10** Two shapes in $U_n$ share boundary if and only if they do not overlap, nor one contains the other, but, there exist two co-equal segments, one from each shape, of which the boundaries overlap (or one contains the other) in $U_{n-1}$.

Then, we say that two shapes are disjoint only if they do not share boundary (nor overlap or one contains the other).

**Definition 3.11** Two shapes that do not overlap, nor share boundary and of which one does not contain the other, are said to be disjoint.
Note that if two segments are not co-equal, they are, necessarily, disjoint. The relations contain, overlap, share boundary and disjoint remain mutually exclusive. With the exception of contain, all relations are anti-reflexive, symmetric and non-transitive. Figure 3.11 illustrates these geometric relations on pairs of shapes in $U^3$.

Figure 3.12 illustrates two special cases: Given two segments $a$ and $b$, Figure 3.12(a) illustrates that the condition $B[a] \cdot B[b] \neq 0$ is not a sufficient, but only a necessary condition for two segments to share boundary; given two shapes that share boundary, Figure 3.12(b) illustrates that, it is not necessary for the two boundaries to overlap, instead the boundary of one shape may contain the boundary of the other shape.

### 3.2.3 Maximal Shapes

The maximal representation of shapes is a canonical representation of shapes as composed of maximal segments.

**Definition 3.12** A segment $a_i (i \leq m)$ in the representation $\{a_1, \ldots, a_m\}$ of a shape $a$ in $U_n$ is denoted a maximal segment if it is disjoint with all segments $a_j (j \leq m, j \neq i)$.
in the representation, and its boundary \(B(a_i)\) is represented as a set of simple boundaries, each of which is a maximal shape in \(U_{n-1}\).\footnote{Within the literature, it is commonly understood that the boundary of a maximal segment is itself a maximal shape. We posit that the boundary of a maximal segment is represented as a set of simple boundaries, each of which is a maximal shape. This additional requirement serves the representation of shapes and shape arithmetic, algorithmically. Also, see Section 4.1.1.}

**Definition 3.13** A shape \(a \{a_1, \ldots, a_m\}\) is termed **maximal** if it is represented as a set of maximal segments \(a_1, \ldots, a_m\).

Using Theorem 3.8 and Theorem 3.11, we posit that the maximal representation is a canonical representation.

A point in the representation of a shape of points is maximal if it is unique in the representation. In \(U_1\), two co-equal line segments \(l_1\) and \(l_2\) are disjoint if either \(\text{head}[l_1] \leq c \text{tail}[l_2]\) and \(\text{head}[l_1] \neq \text{tail}[l_2]\), or \(\text{head}[l_2] \leq c \text{tail}[l_1]\) and \(\text{head}[l_2] \neq \text{tail}[l_1]\). Two line segments with different co-descriptors are necessarily disjoint, even if they intersect.

The canonical representation is particularly advantageous when comparing two shapes for equality or for the subshape relation:
Definition 3.14 A shape \(a\{a_1, \ldots, a_m\}\) in \(U_n\) is embedded in a maximal shape \(b\{b_1, \ldots, b_{m'}\}\) in \(U_n\) if and only if every segment \(a_i\) (\(i \leq m\)) is embedded in a maximal segment \(b_j\) (\(j \leq m'\)).

A constructive definition of embedding can be derived from Lemma 5.8 in Section 5.2.3.

Lemma 3.12 The boundary of a maximal shape \(a\{a_1, \ldots, a_m\}\) equals the sum of the boundaries \(B[a_i]\) (\(i \leq m\));
\[
\forall \text{ maximal } a\{a_1, \ldots, a_m\} \in U_n : B[a] = B[a_1] + \ldots + B[a_m].
\]

Proof: First, consider a co-equal maximal shape \(a\{a_1, \ldots, a_m\}\). The maximal segments \(a_1, \ldots, a_m\) do not share boundary, that is \(B[a_i] \cdot B[a_j] = 0\) for all \(i, j\) with \(i \neq j\). It follows that \(B[a] = B[a_1] \oplus \ldots \oplus B[a_m] = B[a_1] + \ldots + B[a_m]\). In the case of a shape \(a\) with non co-equal maximal segments \(a_i\) and \(a_j\) with \(B[a_i] \cdot B[a_j] = b \neq 0\), the shape \(b\) is a part of the boundary of \(a\). Thus, \(B[a] = B[a_1] + \ldots + B[a_m]\) holds for any maximal shape \(a\{a_1, \ldots, a_m\}\). \(\square\)

Lemma 3.12, contrary to Lemma 3.6, covers all maximal shapes, not only co-equal ones. Furthermore, given a co-equal maximal shape \(a\{a_1, \ldots, a_m\}\), we have that \(B[a_i] \cdot B[a_j] = 0\) for all \(i, j\) with \(i \neq j\). As a result, the set of segments obtained from joining or merging the representations of the shapes \(B[a_i]\) for all \(i \leq m\), is a representation for the shape \(B[a]\) (see Definition 3.7). Note that this representation is not necessarily maximal, because the shapes \(B[a_i]\) may themselves share boundary. Let \(\cup^m\) denote the set operation of merge.

Property Given a co-equal maximal shape \(a\{a_1, \ldots, a_m\}\), the set \(\bigcup B[a_i]\) is a representation for the shape \(B[a]\). This set is also denoted \(\bigcup B[a]\). \(\square\)

Definition 3.15 Given a co-equal shape \(a\{a_1, \ldots, a_m\}\) in \(U_n\), \text{boundary}[a] denotes the result from merging the boundary shapes \(B[a_i]\):
\[
\text{boundary}[a] = \bigcup B[a_i].
\]

Under the maximal representation, the boundary of a segment is represented, not as a set of boundary segments, but as a set of simple boundaries. As such, given a segment \(a\) with \(b_1, \ldots, b_m\) the (maximal) representation of the boundary of \(a\), we write \text{boundary}[a] to denote \(\bigcup b_i\). Similarly, given a co-equal shape \(a\{a_1, \ldots, a_m\}\) in \(U_n\) and given
\{b_1, \ldots, b_m'\} the combined set of simple boundaries of all \(a_i\), we write \(\text{boundary}[a]\) to denote \(\bigcup b_i\). The distinction between the latter statement and Definition 3.15 is only important when we develop the algorithms for the shape operations in Chapter 6. At that time, we are careful to consider the boundary of a shape exactly as it is represented under the maximal representation and to retain any breakup in boundary segments, even at such moments when we discard any grouping into simple boundaries.

From here on, we assume that all shapes are maximal, unless explicitly stated otherwise.
Chapter 4

Geometric Modeling

In Chapter 3, we described the representation of a shape or segment as a pair of mappings, the first from \( U_{n,k} \) to \( \wp(E^k) \), the second from \( U_{n,k} \) to \( U_{n-1,k} \). Under the first, denoted the co-descriptor, each segment in \( U_{n,k} \) \((n \leq k)\) is mapped onto an \( n \)-dimensional hyperplane in \( E^k \). Representationally, the co-descriptor of a segment or point in \( U_{0,k} \) is the \( k \)-tuple of coordinates of the isomorph point in \( E^k \) (see Section 3.1.1) and for a segment in \( U_{n,k} \) corresponds to the equation of the carrier hyperplane in \( E^k \). The co-descriptor relation is an equivalence relation that partitions a shape into a set of co-equal classes of segments or shapes. The second mapping specifies the boundary of a shape in \( U_{n,k} \) as a shape in \( U_{n-1,k} \). By recursion, the boundary mapping relates a shape in \( U_{n,k} \) with a set of shapes in \( U_{0,k} \). Ultimately, the representation of a shape or segment in \( U_{n,k} \) is a mapping to \( \wp(E^k) \). The maximal representation is a canonical boundary representation that specifies the decomposition of a shape into disjoint segments as well as the decomposition of the segments’ boundaries into maximal, simple boundaries (see Section 3.2.3).
In this chapter, we complete the representation of a shape in $U_{n,k}$ (in particular, $U_{n,3}$) in an algorithmic sense, establish an isomorphism between shapes in $U_{n,k}$ and $n$-dimensional point sets in $E^k$ (as referred to in Section 3.1.2, amongst others), and look at solid modeling for a partial evaluation of the algebraic shape model and its geometric representation. Since, most often, we deal with shapes in a 3-dimensional space, we use $U_n$ to denote $U_{n,3}$, when appropriate. Otherwise, we write $U_{n,k}$ specifically.

4.1 Shapes

A maximal shape is represented as a set of disjoint or maximal segments; the representation of a maximal segment consists of a co-descriptor and a boundary; a boundary shape consists of a set of simple boundaries, each of which is represented as a maximal shape. Both the decomposition of a maximal shape into segments and the decomposition of a boundary shape into simple boundaries are unique. We impose a total order on the segments of a shape according to their co-descriptors and, for co-equal segments, according to their boundaries.

**Definition 4.1** Given two segments $a$ and $b$ in $U_{n,k}$, we have that $a \leq b$ if $\text{co}[a] < \text{co}[b]$ or $\text{co}[a] = \text{co}[b] \land \text{B}[a] \leq \text{B}[b]$.

We impose a total order on shapes according to their constituent segments.

**Definition 4.2** Given two shapes $a \{a_1, \ldots, a_m\}$ and $b \{b_1, \ldots, b_{m'}\}$ in $U_{n,k}$, we have that $a \leq b$ if there exists $j \leq \min(m, m')$, such that $a_i = b_i$ for all $i < j$ and $a_j < b_j$, or, if $m \leq m'$ and $a_i = b_i$ for all $i \leq m$.

The same total order applies to the simple boundaries that define a segment. The expressions $\text{B}[a] \leq \text{B}[b]$ and $\text{boundary}[a] \leq \text{boundary}[b]$ are equivalent. In the particular case of two disjoint segments $a$ and $b$, it suffices to compare the first segment from $\text{boundary}[a]$ and $\text{boundary}[b]$. That is, if $\text{boundary}[a]$ corresponds to the ordered set $\{a_1, \ldots, a_m\}$ and $\text{boundary}[b]$ to $\{b_1, \ldots, b_{m'}\}$, then, $\text{B}[a] \leq \text{B}[b]$ if $a_1 \leq b_1$. The total order on the co-descriptors of shapes in $U_{n,k}$ remains to be defined.
4.1 Shapes

4.1.1 Representation

Assertion 3.2 specifies that there exist two neighborhoods of a boundary segment, one of which is a part of the defined shape while the other is disjoint from the defined shape. Using the isomorphism between $U_{n,3}$ and $\mathcal{P}(E^3)$ (see Section 4.2.2), we can define an orientation with respect to a shape or segment that distinguishes the direction of either neighborhood with respect to this (boundary) segment. Such a distinction is necessary to allow for local manipulation of a shape or its boundary. We include this information in our representation, also, for reasons of computational efficiency (see Section 8.1).1

In the sequel we use vectors to represent points, lines and planes in $E^3$. That is, given a point $p$ in $U_0$, the position vector $\vec{p}$ denotes the co-descriptor point in $E^3$. The vectorial representation of a line or plane in $E^3$ that is the co-descriptor of a line segment in $U_1$, respectively, a plane segment in $U_2$, is given below. The relation $\leq$ applied to vectors compares the lexicographical ordering of their coordinates.

A point or segment in $U_{0,k}$ has no boundary. Thus, a segment in $U_{0,k}$ is represented by the $k$-tuple of coordinates that defines the co-descriptor point in $E^k$. We define a total order on points in $U_{0,k}$ corresponding the lexicographical ordering of $k$-tuples or points in $E^k$. This order relation is denoted $\leq c$. Two segments in $U_{0,k}$ are identical if their co-descriptors are equal, and are disjoint, otherwise.

**Definition 4.3** Given two points $p$ and $q$ in $U_{0,3}$ with respective co-descriptor points $\vec{p}$ and $\vec{q}$ in $E^3$, we have that $p \leq c q$ if and only if $\vec{p} \leq c \vec{q}$.

A segment in $U_{1,k}$ has a boundary given by two points, endpoints to the line segment. The co-descriptor of a line segment corresponds to the equation of the infinite line in $E^k$ that carries the segment. A line in $E^3$ is uniquely represented by its Plücker coordinates (Brand, 1947). This is a 6-tuple of which the first 3 components represent the direction vector of the line and the last 3 components represent its moment vector about the origin.

---

1. It is also included in most boundary representations, e.g., the winged-edge (Mäntylä, 1988) and star-edge (Karasick, 1989) representation.
We define the following normalization of a vector:

**Definition 4.4** The norm of a vector $\vec{v}$, denoted $\|\vec{v}\|$, equals $\frac{\vec{v}}{|\vec{v}|}$, if $\vec{v} > 0$, and $-\frac{\vec{v}}{|\vec{v}|}$ if $\vec{v} < 0$.

That is, the norm of a vector is a normalization not only of the length of the vector, but also of its sign, i.e., direction. ²

**Property** $|\vec{v}| = \|-\vec{v}\| \geq 0$. □

Let $p$ and $q$ denote the endpoints of a line segment $l$, with $p \leq q$; we write $p = \text{tail}[l]$ and $q = \text{head}[l]$. Let $\vec{p}$ and $\vec{q}$ represent the points in $E^3$. The direction vector of the carrier of $l$ is the vector $\vec{pq} = \vec{q} - \vec{p}$ and the moment vector equals $\vec{p} \times \vec{q}$. We denote $\vec{d}_l$ the normalized direction vector and write $\vec{d}_l = \|\vec{pq}\|$. We denote $\vec{m}_l$ the (normalized) moment vector and write $\vec{m}_l = \pm \frac{\vec{p} \times \vec{q}}{|\vec{pq}|}$, if $\vec{p} \leq \vec{q}$, and $\vec{m}_l = \frac{\vec{q} \times \vec{p}}{|\vec{pq}|} = \frac{\vec{p} \times \vec{q}}{|\vec{pq}|}$, otherwise (see Figure 4.1).

**Definition 4.5** Given a line segment $l$ with endpoints $p$ and $q$, the direction vector, $\vec{d}_l$, of the carrier of $l$ equals the vector $\|\vec{p} - \vec{q}\|$ and the moment vector, $\vec{m}_l$, of the carrier of $l$ equals the vector $\frac{\vec{p} \times \vec{q}}{|\vec{pq}|}$, if $p \leq q$, and $\frac{\vec{q} \times \vec{p}}{|\vec{pq}|}$, otherwise.

The co-descriptor of a line segment $l$ with direction vector $\vec{d}_l$ ($d^x, d^y, d^z$) and moment vector $\vec{m}_l$ ($m^x, m^y, m^z$) is the Plücker coordinates 6-tuple $(d^x, d^y, d^z, m^x, m^y, m^z)$. The

2. In this respect, our definition differs from the generally accepted definition for the norm of a vector.
4.1 Shapes

The equation of the line with direction vector $\vec{d}_l$ and moment vector $\vec{m}_l$ equals $\vec{x} \times \vec{d}_l = \vec{m}_l$, or in parametric form $\vec{x} = \alpha \vec{d}_l - \vec{d}_l \times \vec{m}_l$ (Brand, 1947).

We define a total order on the co-descriptor of a line segment corresponding to the lexicographical ordering of the Plücker coordinate tuples. That is, given two segments $l$ and $l'$ in $U_1$, with direction vectors $\vec{d}_l$ and $\vec{d}_l'$ and moment vectors $\vec{m}_l$ and $\vec{m}_l'$, $\text{co}[l] \leq \text{co}[l']$ if $\vec{d}_l \times \vec{m}_l < \vec{d}_l' \times \vec{m}_l'$ or $\vec{d}_l = \vec{d}_l' \wedge \vec{m}_l \leq \vec{m}_l'$. Given two segments $l$ and $l'$ in $U_1$, we have that $l \leq l'$ if $\text{co}[l] < \text{co}[l']$ or $\text{co}[l] = \text{co}[l'] \wedge \text{tail}[l] < \text{tail}[l']$ or $\text{co}[l] = \text{co}[l'] \wedge \text{tail}[l] = \text{tail}[l'] \wedge \text{head}[l] \leq_c \text{head}[l']$.

Two line segments $l$ and $l'$ in $U_1$ are coplanar if the sum of the scalar products of each direction vector with the other moment vector (i.e., of the other line segment) equals 0, that is, if $\vec{d}_l \cdot \vec{m}_{l'} + \vec{d}_{l'} \cdot \vec{m}_l = 0$. Two line segments $l$ and $l'$ are parallel if their direction vectors are identical, that is, if $\vec{d}_l = \vec{d}_{l'}$. Two line segments $l$ and $l'$ are perpendicular if the scalar product of the direction vectors equals 0, that is, if $\vec{d}_l \cdot \vec{d}_{l'} = 0$. We have that the carriers of two line segments $l$ and $l'$ in $U_{1,k}$ intersect if $l$ and $l'$ are coplanar but are not parallel. We say that the two line segments $l$ and $l'$ intersect if they are coplanar, not parallel and the point $p$ in $U_{0,k}$, isomorphic to the point of intersection of the carriers of $l$ and $l'$, satisfies the inequalities

$$\text{tail}[l] \leq_c p \leq_c \text{head}[l] \wedge \text{tail}[l'] \leq_c p \leq_c \text{head}[l'].$$

Note that we include the endpoints as acceptable intersection points.

We elaborate on the notion of intersection in some detail.

Consider a neighborhood $\Delta(p)$ of a point $p$. Let $\vec{d}$ denote the direction vector of the carrier of $\Delta(p)$. The boundary $B[\Delta(p)]$ of $\Delta(p)$ consists of two points, one of which is $p$; let $q$ denote the other endpoint. The direction vector $\vec{d}$ defines an orientation of a neighborhood $\Delta(p)$ with respect to $p$: We say that the inside of $\Delta(p)$ is oriented positively with respect to $p$ if $\frac{\vec{p}q}{|\vec{pq}|} = \vec{d}$ and negatively, otherwise. If $p = \text{tail}[\Delta(p)]$, then, we have that $p \leq q$ and $\vec{d} = \frac{\vec{pq}}{|\vec{pq}|}$; otherwise, if $p = \text{head}[\Delta(p)]$, then, we have that $\vec{d} = -\frac{\vec{pq}}{|\vec{pq}|}$. Thus, the inside of any line segment $l$ is oriented positively with respect to $\text{tail}[l]$ and negatively with respect to $\text{head}[l]$.

\[\text{equation of the line with direction vector } \vec{d}_l \text{ and moment vector } \vec{m}_l \text{ equals } \vec{x} \times \vec{d}_l = \vec{m}_l, \text{ or in parametric form } \vec{x} = \alpha \vec{d}_l - \vec{d}_l \times \vec{m}_l (\text{Brand}, 1947).\]
A plane segment in $U_{2,k}$ has a boundary shape in $U_{1,k}$ consisting of one or more simple boundaries, one of which is an outer boundary while the other are inner boundaries (see Section 3.2.1). The co-descriptor of a plane segment corresponds to the equation of the infinite plane in $E^k$ that carries the segment. In $E^3$, given a plane $f$ with unit normal vector $\vec{n}_f$ and normal distance $d_f$ from the origin, the equation of the plane $f$ equals $\vec{x} \cdot \vec{n}_f + d_f = 0$. Given a plane segment $f$ and two non-parallel boundary line segments of $f$ with direction vectors $\vec{d}_l$ and $\vec{d}_l'$, the normalized normal vector $\vec{n}_f$ of the carrier of $f$ equals the norm of the vector product of these direction vectors, $\vec{n}_f = \|\vec{d}_l \times \vec{d}_l\|$ (see Figure 4.2). The corresponding normal distance $d_f$ follows from the equation of the carrier plane, that is, given any point $\vec{p}$ on the carrier of $f$, it holds that $d_f = -\vec{p} \cdot \vec{n}_f$. We have that $\alpha \vec{d}_l - \vec{d}_l \times \vec{m}_l$ represents a point on the carrier of $l$ and, therefore, on the carrier of $f$, for any scalar $\alpha$. If we choose $\alpha = 0$, we have that $d_f = (\vec{d}_l \times \vec{m}_l) \cdot \vec{n}_f$.

**Definition 4.6** Given a plane segment $f$ in $U_2$ with boundary $B(f) \{l_1, \ldots, l_n\}$, the normal vector, $\vec{n}_f$, of the carrier of $f$ equals $\|\vec{d}_i \times \vec{d}_j\|$ for any $i, j$ with $i \neq j$ and $\vec{d}_i \neq \vec{d}_j$, and the normal distance, $d_f$, of the carrier of $f$ equals $(\vec{d}_i \times \vec{m}_l) \cdot \vec{n}_f$ for any $i \leq n$.

The co-descriptor of a plane segment $f$ with normal vector $\vec{n}_f$ ($n_x^f$, $n_y^f$, $n_z^f$) is the 4-tuple ($n_x^f$, $n_y^f$, $n_z^f$, $d_f$).

We define a total order on the co-descriptor of a plane segment in $U_2$ corresponding to the lexicographical ordering of the 4-tuples. That is, given two segments $f$ and $g$ in $U_2$, with normal vectors $\vec{n}_f$ and $\vec{n}_g$ and normal distances $d_f$ and $d_g$, $co[f] \leq co[g]$ if $\vec{n}_f < \vec{n}_g$ or

**Figure 4.2** A plane segment $f$ with boundary line segments $l$ and $l'$, the normal vector and normal distance of the carrier of $f$. 

\[
\vec{n}_f = \|\vec{d}_l \times \vec{d}_l\|
\]
\[
d_f = (\vec{d}_l \times \vec{m}_l) \cdot \vec{n}_f
\]
Two plane segments $f$ and $g$ are coplanar if their co-descriptors are identical, that is, if $\vec{n}_f = \vec{n}_g$ and $d_f = d_g$. Two plane segments $f$ and $g$ are parallel if their normal vectors are identical, that is, if $\vec{n}_f = \vec{n}_g$. Two line segments $f$ and $g$ are perpendicular if the scalar product of the normal vectors equals $0$, that is, if $\vec{n}_f \cdot \vec{n}_g = 0$.

We have that the carriers of two plane segments $f$ and $g$ in $U_2$ intersect if $f$ and $g$ are not parallel, and when they do, they do so in a line. The conditions for two plane segments to intersect is developed in Section 6.2 (see Corollary 6.2).

Under an isomorphism between shapes in $U_n$ and point sets in $E^3$ (see Section 4.2.2), consider a segment $a$ in $U_n$ and its isomorph point set $a^*$, as well as a boundary segment $b$ of $a$ and its isomorph point set $b^*$. Let $\setminus$ denote the Boolean set operation of difference (on point sets).

**Definition 4.7** A vector $\vec{v}$ indicates the inside of a segment $a$ in $U_n$ with respect to a boundary segment $b$ of $a$ if there exists a point in $E^3$, denoted $\vec{p}$, that lies in $b^*$, and an infinitesimal scalar $\varepsilon$ such that $\vec{p} + \varepsilon \vec{v}$ denotes a point in $a^* \setminus b^*$.

Consider a neighborhood $\Delta(l)$ of a line segment $l$. Let $\vec{n}$ denote the normal vector of the carrier of $\Delta(l)$. This vector defines an orientation of $\Delta(l)$ with respect to $l$ as follows.

**Definition 4.8** The inside of a neighborhood $\Delta(l)$ of a line segment $l$ is oriented positively with respect to $l$ if the vector product $\vec{n} \times \vec{l}$ of the normal vector of the carrier of $\Delta(l)$ and the direction vector of the carrier of $l$, indicates the inside of $\Delta(l)$ with respect to $l$, and negatively, otherwise.

We denote $\text{inside}[l, \Delta(l)]$ the orientation of the inside of $\Delta(l)$ with respect to $l$. That is, $\text{inside}[l, \Delta(l)] = +1$ ($> 0$) if $\vec{n} \times \vec{l}$ indicates the inside of $\Delta(l)$ with respect to $l$, and $\text{inside}[l, \Delta(l)] = -1$ ($< 0$), otherwise (Figure 4.3 illustrates both cases). We will write $\text{inside}[l]$, shortly, if the particular neighborhood $\Delta(l)$ of $l$ is obvious from the context. Note that the direction of $\vec{n} \times \vec{l}$ is only dependent on the carriers of $l$ and $\Delta(l)$, not on their boundaries. Therefore, $\text{inside}[l, \Delta(l)] = \text{inside}[l', \Delta(l')]$ for any $l \leq l'$ or $l \leq l'$ and for any $\Delta(l) \leq \Delta(l)$ or $\Delta(l) \leq \Delta(l')$.

A volume segment in $U_{3,k}$ has a boundary shape in $U_{2,k}$ consisting of one or more simple boundaries, one of which is an outer boundary while the other are inner boundaries (see...
Section 3.2.1). The co-descriptor of a volume segment corresponds to the equation of the infinite volume in $E^k$ that carries the segment. In $E^3$, only one infinite volume exists, namely, the space $E^3$. Thus, the co-descriptor is identical for all segments or shapes in $U_3$ and, therefore, is redundant in the representation of a volume segment in $U_3$. Furthermore, shapes in $U_3$ do not intersect.

Consider a neighborhood $\Delta(f)$ of a plane segment $f$. Let $\vec{n}_f$ denote the normal vector of the carrier of $f$. We define an orientation of $\Delta(f)$ with respect to $f$ as follows.

**Definition 4.9** The inside of a neighborhood $\Delta(f)$ of a plane segment $f$ is oriented positively with respect to $f$ if the normal vector $\vec{n}_f$ of the carrier of $f$ indicates the inside of $\Delta(f)$ with respect to $f$, and negatively, otherwise.

Again, we denote $\text{inside}[f, \Delta(f)]$ the orientation of the inside of $\Delta(f)$ with respect to $f$. That is, $\text{inside}[f, \Delta(f)] = +1$ if $\vec{n}_f$ indicates the inside of $\Delta(f)$ with respect to $f$, and $\text{inside}[l, \Delta(l)] = 1$, otherwise. Note that the direction of $\vec{n}_f$ is only dependent on the carrier of $f$, not on the boundaries of $f$ or $\Delta(f)$. Therefore, $\text{inside}[f, \Delta(f)] = \text{inside}[g, \Delta(g)]$ for any $g \leq f$ or $f \leq g$ and for any $\Delta(g)$ in $U_3$. We will write $\text{inside}[f]$, for brevity, whenever the particular neighborhood $\Delta(f)$ of $f$ is obvious from the context.

When the orientation of the inside of a shape with respect to a boundary segment makes an integral part of the representation of the boundary segment, the maximality of two co-equal, boundary segments that share boundary becomes dependent upon whether they have equal or opposite inside orientations. The two boundary segments can only be combined when they have equal inside orientations. However, the maximal representation of the boundary
4.1 Shapes

of a segment (Definition 3.12) specifies a partitioning of the boundary segments into simple boundaries. If the orientations of the inside of a shape are opposite with respect to two co-equal, boundary segments that share boundary, these segments necessarily belong to different simple boundaries as is illustrated in Figure 4.4. Therefore, the augmentation is compatible with the defined maximal representation.

4.1.2 Cross-Algebra Operations

In Chapter 2, we described the operations of sum, product, difference and symmetric difference on shapes within the same algebra. The result of each of these operations is a shape within the same algebra and, given two segments, each of these operations is non-trivial only if the two segments are co-equal (actually, non-disjoint). On the contrary, the operation of intersection, described above, applies to two segments or shapes within the same algebra $U_n$ that are not co-equal, and results in a shape in the algebra $U_{n-1}$.

The algorithmic approach to the algebraic operations of sum, product, difference and symmetric difference, as developed in Chapter 5, uses a classification of the boundary segments of one shape with respect to another shape, both in $U_n$. In order to compare or classify a segment in $U_{n-1}$ with a shape in $U_n$, we extend the notion of intersection among shapes, from shapes within the same algebra, to shapes in different algebras (but not unrelated). Then, the operation of intersection also applies to two segments or shapes, denoted $a$ and $b$, where $a$ is a shape in the algebra $U_n$ and $b$ is a shape in the algebra $U_{n-1}$ and the carrier of $b$ lies within the carrier of $a$. The result of this operation is a shape in $U_{n-1}$ that is a part of $b$. We refer to Chapter 5 through Chapter 7 for its development.

Figure 4.4 Two co-equal boundary line segments $l$ and $l'$ of a plane segment $f$, that share boundary and have respective opposite orientations of the inside of $f$. 

\[ \text{inside}[l,f] \neq \text{inside}[l',g] \]
4.2 Solid Modeling

It is obvious that we can draw a parallel between shape arithmetic using a maximal representation on volume segments, and Boolean set operations on solids under a boundary representation. In Chapter 2 we developed an algebraic model for shapes and shape arithmetic based on the part relation; in Chapter 3, we extended this model with a boundary representation in terms of maximal segments. At this time, it is appropriate to compare our approach for shapes and shape arithmetic to the established theories and practices in the field of solid modeling. The following is a discussion of the similarities and dissimilarities between shapes in $U_{3,k}$ and solids in $E^k$, and between shape arithmetic and the Boolean set operations on solids. We define an isomorphism between $U_{n,k}$ and a subset of $\wp(E^k)$ and show that the regularized set operations union, intersection and difference are equivalent to the shape operations sum, product and difference, under the isomorphism. Finally, we compare the expressive power of both models. We refer to Mäntylä (1988) and Hoffmann (1989a) for an introduction to geometric and solid modeling.

4.2.1 Solids

Mäntylä (1988) specifies two mathematical models for defining solids. One uses concepts from point-set topology, the other from algebraic topology, thereby adopting a surface-based model. In the former, a solid is defined as a bounded, closed subset of the Euclidean space $(E^3)$, with additional constraints of rigidity, regularity and representational finiteness. In the latter, a solid is defined as bounded by a plane model of 2-manifolds, i.e., abstractions of closed surfaces, with additional realizability constraints.

A regular point set is a point set that equals the closure of the interior of itself. Given a point set $S$ in $E^3$, let $c(S)$ denote the closure of $S$ and $i(S)$ the interior of $S$. Then, the regularization of a point set $S$, denoted $r(S)$, is defined by $r(S) = c(i(S))$, and sets that satisfy $r(S) = S$ are said to be regular (Mäntylä, 1988). Informally, a regular point set in $E^3$ is a proper 3-dimensional point set, without erroneous (“dangling”) lower dimensional subsets. Figure 4.5 illustrates a non-regular point set in $E^3$ and its regularized counterpart.
In order to demonstrate a correspondence between shapes in \( U_{n,k} \) and point sets in \( E^k \), we extend the notions of closure, interior and regularization to \( n \)-dimensional point sets in \( E^k \) \((n \leq k)\). We consider an \( n \)-dimensional point set in \( E^k \) to be a subset of an \( n \)-dimensional hyperplane in \( E^k \).

**Definition 4.10** Given a point set \( S \) in \( E^k \) that is a subset of an \( n \)-dimensional hyperplane \( H \) and a continuous transformation \( f \) that maps \( H \) onto \( E^n \), we say that \( S \) is \( n \)-regular in \( E^k \) if and only if \( f(S) \) is regular in \( E^n \), i.e., \( f(S) \equiv r(f(S)) \).

The Boolean set operations union, intersection and difference do not necessarily preserve regularity. Figure 4.6 illustrates two regular point sets in \( E^3 \) that “touch”, that is, they share a “face” or part thereof, such that the intersection of the two sets contains a “dangling face”, and the difference of one set with the other is a point set that is no longer closed and regular. To remedy this, regularized set operations are defined that are regularity-preserving variants of the Boolean set operations (Mäntylä, 1988). Let \( \cup, \cap, \text{ and } \setminus \) denote the Boolean set operations union, intersection and difference, respectively. Then, the regularized set operations \( \cup^*, \cap^* \) and \( \setminus^* \) are defined as

\[
\begin{align*}
S \cup^* T &= r(S \cup T) \\
S \cap^* T &= r(S \cap T) \\
S \setminus^* T &= r(S \setminus T)
\end{align*}
\]
A 2-manifold is a topological space where every point has a neighborhood topologically equivalent to an open disk of $E^2$ (Mäntylä, 1988). Figure 4.7(a) illustrates three points with neighborhoods topologically equivalent to an open disk of $E^2$ on a 2-manifold surface. However, not all solids, even regular, have 2-manifold surfaces, e.g., the solid in Figure 4.7(b) is nonmanifold even though the composing solids have 2-manifold surfaces. In such cases, an indirect representation by 2-manifolds may be considered that “ignores” the exceptional points and line segments (Mäntylä, 1988). Yet, the problem of manifold versus nonmanifold goes further than just the mathematical model. Most boundary representations schemes, such as the winged-edge (Mäntylä, 1988) or star-edge (Karasick, 1988), assume that solids are bounded by 2-manifold surfaces. This allows one to sort all the edges leaving a single vertex, given the order of the faces or loops they belong to, about the vertex. Yet,
the result of a Boolean set operation, even regularized, on two solids bounded by 2-manifold surfaces does not always yield a solid that is bounded by a 2-manifold surface.

4.2 Solid Modeling

4.2.2 Isomorphisms

A volume segment or, in general, a maximal shape in \( U_{n,k} \) is defined by a set of disjoint boundaries, each of which constitutes a maximal shape in \( U_{n-1,k} \). However, these boundary shapes do not constitute part of the defined shape in \( U_{n,k} \). Nor does there exist an absolute complement to a shape, to which these boundaries form part of. These boundaries only serve to define the shape in \( U_{n,k} \).

As a result, there is no inherent notion of an open or closed shape and, in principle, there exists no equivalent point set in \( E^k \) to a shape in \( U_{3,k} \). Even then, it is possible to define an isomorphism between \( U_{n,k} \) and a subset of \( \wp(E^k) \) that maps each shape or element in \( U_{n,k} \) to a point set in \( E^k \), and to define “regularized” set operations of union, intersection and difference that are equivalent to the shape operations of sum, product and difference, under the isomorphism.

**Theorem 4.1** There exists an isomorphism between \( U_{n,k} \) \((1 \leq n \leq k)\) and a subset of \( \wp(E^k) \) that maps each shape \( x \) in \( U_{n,k} \) to an \( n \)-regular point set \( x^* \) in \( E^k \).

**Proof:** In Section 3.1.1, we have shown that an isomorphism exists between \( U_{0,k} \) and \( E^k \) that maps each point \( p \) in \( U_{0,k} \) to a point \( p^* \) in \( E^k \).

Consider a line segment \( l \) in \( U_{1,k} \) with boundary points \( p \) and \( q \). The isomorph points \( p^* \) and \( q^* \) in \( E^k \) lie on the carrier of \( l \). Let \( l^* \) denote the subset of the carrier of \( l \) that is bounded by \( p^* \) and \( q^* \), with both endpoints included. It follows that \( l^* \) is a 1-regular point set.

Given a shape \( l \) \((l_1, \ldots, l_m)\) in \( U_{1,k} \), let \( l^* \) denote the union of the point sets \( l_1^*, \ldots, l_m^* \) in \( E^k \).

Then, for each shape \( l \) in \( U_{1,k} \), there exists exactly one point set \( l^* \) element of \( \wp(E^k) \) and no two shapes \( l_1 \) and \( l_2 \) in \( U_{1,k} \) map onto the same point set in \( E^k \). Thus, there exists an isomorphism between \( U_{1,k} \) and a subset of \( \wp(E^k) \) that maps each shape \( l \) in \( U_{1,k} \) to a 1-regular point set \( l^* \) in \( E^k \).
Similarly, consider a plane segment \( f \) in \( U_{2,k} \) with boundary shape \( l = B[f] \). The isomorph point set \( l^* \) in \( E^k \) forms a closed polygon that lies in the carrier of \( f \). Let \( f^* \) denote the subset of the carrier of \( l \) that is enclosed by the polygon \( l^* \), with \( l^* \) included. It follows that \( f^* \) is a 2-regular point set. Given a shape \( g \ (g_1, \ldots, g_m) \) in \( U_{2,k} \), let \( g^* \) denote the union of the point sets \( g_1^*, \ldots, g_m^* \) in \( E^k \). Then, similarly to \( U_{1,k} \), there exists an isomorphism between \( U_{2,k} \) and a subset of \( \wp(E^k) \) that maps each shape \( f \) in \( U_{2,k} \) to a 2-regular point set \( f^* \) in \( E^k \).

Also, similarly, there exists an isomorphism between \( U_{3,k} \) and a subset of \( \wp(E^k) \) that maps each shape \( s \) in \( U_{3,k} \) to a 3-regular point set \( s^* \) in \( E^k \), where \( s^* \) denotes the subset of the carrier of \( s \) that is enclosed by the polyhedron \( f^* \), with \( f^* \) included, and where \( f^* \) is the 2-regular point set isomorphic to the boundary of \( s \).

We conclude, by induction, that an isomorphism exists between \( U_{n,k} \) and a subset of \( \wp(E^k) \) that maps each shape \( x \) in \( U_{n,k} \) to an \( n \)-regular point set \( x^* \) in \( E^k \), where \( x^* \) denotes the subset of the carrier of \( x \) that is enclosed by the \((n-1)\)-dimensional polyhedron \( y^* \), with \( y^* \) included, and where \( y^* \) is the \((n-1)\)-regular point set isomorphic to the boundary shape \( y = B[x] \) in \( U_{n-1,k} \). It follows from the construction, that the respective boundary operations on shapes and regular point sets are equivalent under the isomorphism.

Given a point \( p \) in \( U_{0,k} \), if we consider \( p^* \) to denote the singleton point set in \( E^k \) that is a 0-regular point set, then the theorem also holds for \( n = 0 \).

Consider the \( n \)-regularized set operations \( \cup^*, \cap^*, \) and \( \setminus^* \). Then, given two shapes \( a \) and \( b \) in \( U_{n,k} \) and the sum \( c = a + b \), it holds that \( c^* = a^* \cup^* b^* \), and similarly for the shape operation of product and \( \cap^* \) as well as the shape operation of difference and \( \setminus^* \). Thus, the \( n \)-regularized set operations of union, intersection and difference are equivalent to the shape operations of sum, product and difference, under the isomorphism of Theorem 4.1. Similarly, the subset relation on \( n \)-regularized point sets is equivalent to the part relation on shapes, under the isomorphism. These equivalences are shown (implicitly) in Chapter 5, as we employ the same classification to define the results of the arithmetic operations on shapes, as are used to determine the results of the regularized set operations on point sets.
A regular point set is a set that equals the closure of the interior of itself, that is a closed set. For the sake of argument, consider an alternative regularization of a point set as the interior of the closure of the set. That is, \( r^\dagger(S) = i(c(S)) \) and \( r^\dagger(S) \) is an open set. Again, we extend the notions of closure, interior and regularization to \( n \)-dimensional point sets in \( E^k \) \((n \leq k)\) as exemplified in Section 4.2.1.

This is an alternative definition of regular sets as open sets. Also, consider the alternatively \( n \)-regularized set operations \( \cup^\dagger \), \( \cap^\dagger \), and \( \setminus^\dagger \), respectively, defined as

\[
S \cup^\dagger T = r^\dagger(S \cup T) \\
S \cap^\dagger T = r^\dagger(S \cap T) \\
S \setminus^\dagger T = r^\dagger(S \setminus T)
\]

**Corollary 4.2** There exists an isomorphism between \( U_{n,k}(1 \leq n \leq k) \) and a subset of \( \wp(E^k) \) that maps each shape \( x \) in \( U_{n,k} \) to an \( n \)-regular\( ^\dagger \) point set \( x^\dagger \) in \( E^k \).

**Proof:** Given a shape \( x \) in \( U_{n,k} \), let \( x^\dagger \) denote the subset of the carrier of \( x \) in \( E^k \) that is enclosed by the \((n-1)\)-dimensional polyhedron \( y^\dagger \), with \( y^\dagger \) excluded, and where \( y^\dagger \) is the \((n-1)\)-regular point set isomorphic to the boundary shape \( y = B[x] \) in \( U_{n-1,k} \). It follows that \( x^\dagger \) is an \( n \)-regular\( ^\dagger \) point set. The proof is similar to the proof of Theorem 4.1.

It follows that \( x^\dagger = i(x^\ast) = x^\ast \setminus y^\ast \). Given two shapes \( a \) and \( b \) in \( U_{n,k} \) and the sum \( c = a + b \), it holds that \( c^\dagger = a^\dagger \cup^\dagger b^\dagger \), and similarly for the shape operation of product and \( \cap^\dagger \) as well as the shape operation of difference and \( \setminus^\dagger \). Thus, the alternatively \( n \)-regularized set operations of union, intersection and difference are equivalent to the shape operations of sum, product and difference, under the isomorphism of Corollary 4.2.

Each of the previous isomorphisms required (alternatively) regularized set operators to correspond to the shape operations of sum, product and difference. The following isomorphism between \( U_{3,3} \) and a subset of \( \wp(E^3) \) preserves the “regularity” of hybrid point sets (half-open, half-closed) in \( E^3 \) under the original Boolean set operations \( \cup, \cap, \) and \( \setminus \). Let \( F_s \subset \text{boundary}[x] \) denote the set \( \{f_1, ..., f_m\} \) of boundary plane segments \( f_i \) with \( \text{inside}[f_i, s] > 0 \), that is, the orientation of the inside of the shape \( s \) with respect to each of these segments \( f_i \) is positive.
Consider any line segment \( l \) that is a part of the boundaries of some segments \( g_j \in \text{boundary}[s] \) and is disjoint with the boundaries of all other segments in \( \text{boundary}[s] \).

That is, there exist plane segments \( g_1, \ldots, g_m' \), with \( g_j \in \text{boundary}[s] \), such that \( l \leq B[g_j] \) for all \( j \leq m' \) and \( l \cdot B[g] = 0 \) for all \( g \in \text{boundary}[s] \), \( g \neq g_j \) \((j \leq m') \). We consider a reference half-plane \( f_r \) with boundary \( l \) that is parallel to the X-axis, unless \( l \) is parallel to the X-axis, in which case we consider \( f_r \) parallel to the Y-axis, such that a vector \( \vec{v} \) indicating the inside of \( f_r \) with respect to \( l \) is positive. Given such a line segment \( l \) and the plane segments \( g_1, \ldots, g_m' \) that have \( l \) as a boundary segment, let \( f_j \) denote the plane segment \( g_i \) \((i \leq m)\) that makes the smallest angle with \( f_r \). Then, let \( L_s \subset \text{boundary}[\text{boundary}[s]] \) denote the set of boundary segments \( l \), with corresponding \( f_j \), for which \( f_j \in F_s \).

Given a boundary line segment \( l \) and the corresponding plane segment \( f_j \), consider the reference vector \( \vec{r}_l \) to be the normalized vector product of the normal vector of \( f_j \) and the direction vector of \( l \), i.e., \( \vec{r}_l = \frac{\vec{n}_{f_j} \times \vec{d}_l}{\|\vec{n}_{f_j} \times \vec{d}_l\|} \). Then, we show that \( l \in L_s \) if and only if \( \text{inside}[f_j] \vec{n}_{f_j} \cdot \vec{r}_l > 0 \), when \( \vec{v}_{f_j} \cdot \vec{r}_l \neq 0 \), or \( \text{inside}[f_j] > 0 \), otherwise. From the Jordan curve theorem, we know that two consecutive plane segments \( g_i \) and \( g_j \), in a clockwise or counterclockwise configuration about \( l \), define a wedge that is either inside or outside with respect to \( s \), and which alternates between inside and outside as one considers consecutive wedges. Given the plane segment \( f_j \) that makes the smallest angle with \( f_r \), the wedge containing \( f_r \) is inside with respect to \( s \), if and only if \( f_j \in F_s \). Therefore, when \( f_j \in F_s \), the reference vector \( \vec{r}_l \) indicates the inside of \( s \) with respect to \( l \). As such, if \( \vec{v}_{f_j} \cdot \vec{r}_l \neq 0 \), that is, the carrier of \( f_j \) and \( f_r \) do not coincide, then, the inside of \( s \) with respect to \( f_j \) is indicated by the vector \( \text{inside}[f_j] \vec{n}_{f_j} \) and, therefore, \( \vec{r}_l \) indicates the inside of \( s \) with respect to \( l \) if and only if \( \text{inside}[f_j] \vec{n}_{f_j} \cdot \vec{r}_l > 0 \) (see Figure 4.8).

Consider a point \( p \) that is an endpoint of a boundary line segment of a boundary plane segment of \( s \). Consider the plane segments \( g_1, \ldots, g_m \), with \( g_i \in \text{boundary}[s] \), such that \( p \) is...
an endpoint of a boundary line segment of $g_i$ for all $i \leq m$ and $p$ is not an endpoint of a boundary line segment of any $g \in \text{boundary}[s]$ with $g \neq g_i$ ($i \leq m$). Consider a reference plane $f_r$ through $p$, parallel to the XY-plane and consider a reference half-line $k_r$ with endpoint $p$, parallel to the X-axis, such that the direction vector $\mathbf{r}_p$ of $k_r$ indicates the inside of the $k_r$ with respect to $p$. Consider the line segments of intersection of $g_i$ ($i \leq m$) and $f_r$ that have $p$ as an endpoint. Let $k_p$ denote the line segment of intersection that makes the smallest angle with $k_r$. Then, $P_s \subset \text{boundary}[\text{boundary}[\text{boundary}[s]]]$ denotes the set of endpoints $p$, with corresponding $k_l$, such that, in the case of a single plane segment $f_p$ corresponding to $k_p$, we have that $f_p \in F_s$ and, otherwise, $k_p \in L_s$. Figure 4.9 illustrates four different cases that may occur.

Let $F_s^\dagger$, $L_s^\dagger$ and $P_s^\dagger$ denote the point sets isomorphic to the shapes $F_s$, $L_s$ and $P_s$, respectively, under the isomorphism of Corollary 4.2. Consider the point set $s^\dagger = s^\dagger \cup F_s^\dagger \cup L_s^\dagger \cup P_s^\dagger$ corresponding to the shape $s$ in $U_3$. We prove that this constitutes an isomorph

4. Let $\mathbf{v}_k$ ($i \leq m$) denote a vector indicating the inside of $k_i$ with respect to $p$, i.e., $\mathbf{v}_k = \text{inside}[p, k_i] \mathbf{a}_k$. Then, the configuration (i.e., ordering) of the line segments $k_i$ is identical to the configuration of the vectors $\mathbf{v}_k$ about $p$. Since the scalar product of two vectors is proportional to the cosine of the angle defined by the two vectors, it suffices to compute the values $\mathbf{v}_k \cdot \mathbf{r}_p$, for $i \leq m$, in order to determine the line segment that makes the smallest angle with $k_r$. 

Figure 4.8 A line segment $l$ is an element of $L_s$ if the normal vector indicating the inside of $s$ with respect to $f_l$ and the reference vector $\mathbf{r}_l$ make an acute angle, i.e., their scalar product is positive and non-zero.
construction of point sets in $E^3$ from shapes in $U^3$ that preserves “regularity” under the Boolean set operations $\cup$, $\cap$, and $\setminus$.

**Lemma 4.3** There exists an isomorphism between $U^3$ and a subset of $\wp(E^3)$ that maps each shape $x$ in $U^3$ to a regular point set $x^\parallel$ in $E^3$.

**Proof:** We show that the result of any of the Boolean set operations union, intersection and difference on two point sets $s^\parallel$ and $t^\parallel$, corresponding to shapes $s$ and $t$ in $U^3$, respectively, is a point set $r^\parallel$ that corresponds to a shape $r$ in $U^3$, under the same construction. In particular, we examine the boundary of $s^\parallel \cup t^\parallel$, $s^\parallel \cap t^\parallel$, and $s^\parallel \setminus t^\parallel$ for each of the cases of coinciding “faces”, “edges” and “vertices”.

Consider a “face” $f^\parallel$ of both $s^\parallel$ and $t^\parallel$. Assume $f^\parallel \subset F_s^\parallel$ as well as $f^\parallel \subset F_t^\parallel$. Then, $f$ is a boundary plane segment of both $s$ and $t$ and the normal vector $\vec{n}_f$ of the carrier of $f$ indicates the insides of both $s$ and $t$ with respect to $f$. We say that $f$ is deemed same-shared with respect to $s$ and $t$. It follows that $f$ is a boundary plane segment of both $s + t$ and $s \cdot t$, and, $\vec{n}_f$ indicates the insides of both shapes with respect to $f$, but, $f$ is not a boundary plane segment of $s - t$ (or, $t - s$). Using the corresponding Boolean set operations, we find that
4.2 Solid Modeling

Let \( f^\dagger \subset (F_s^\dagger \cup F_t^\dagger) \subset (s^\dagger \cup t^\dagger) \), as well as, \( f^\dagger \not\subset (F_s^\dagger \setminus F_t^\dagger) \subset (s^\dagger \setminus t^\dagger) \). Now, assume that \( f^\dagger \not\subset F_s^\dagger \) and \( f^\dagger \not\subset F_t^\dagger \). Again, \( f \) is a boundary segment deemed same-shared with respect to \( s \) and \( t \), but, the normal vector \( \vec{n}_f \) of the carrier of \( f \) indicates the outsides of both \( s \) and \( t \) with respect to \( f \). We have that \( f^\dagger \not\subset (s^\dagger \cup t^\dagger) \), \( f^\dagger \not\subset (s^\dagger \cap t^\dagger) \) and \( f^\dagger \not\subset (s^\dagger \setminus t^\dagger) \). These cases are illustrated in Figure 4.10. The remaining case, when \( f^\dagger \subset F_s^\dagger \) and \( f^\dagger \not\subset F_t^\dagger \), or inversely, is illustrated in Figure 4.11. The derivation is similar as above.

Consider an “edge” \( l^\dagger \) of both \( s^\dagger \) and \( t^\dagger \). If \( l^\dagger \subset L_s^\dagger \), then, we have that the corresponding vector \( \vec{r}_l \) indicates the inside of \( s \) with respect to \( l \). Therefore, \( \vec{r}_l \) indicates the inside of \( s \cdot t \) with respect to \( l \) only, if \( \vec{r}_l \) also indicates the inside of \( t \) with respect to \( l \). That is, if \( l^\dagger \subset L_t^\dagger \), then, \( l^\dagger \subset (L_s^\dagger \cap L_t^\dagger) \subset (s^\dagger \cap t^\dagger) \). We have that \( \vec{r}_l \) always indicates the inside of \( s + t \) with respect to \( l \); \( l^\dagger \subset L_s^\dagger \subset (L_s^\dagger \cup L_t^\dagger) \subset (s^\dagger \cup t^\dagger) \) (see Figure 4.12). Similarly, \( \vec{r}_l \) indicates the inside of \( s - t \) with respect to \( l \), only, if \( \vec{r}_l \) does not indicate the inside of \( t \) with respect to \( l \). That is, if \( l^\dagger \not\subset L_s^\dagger \), then, \( l^\dagger \subset (L_s^\dagger \setminus L_t^\dagger) \subset (s^\dagger \setminus t^\dagger) \). A similar derivation holds if \( l^\dagger \not\subset L_t^\dagger \).

5. We refer to Chapter 5 for a formal exposition: Definition 5.2 and Definition 5.3 define same-shared and opposite-shared boundary segments with respect to two shapes; Lemma 5.4 through Lemma 5.7 define the boundary shapes resulting from each of the operations of sum, product, difference and symmetric difference in terms of the boundary shapes of both operands.
A similar exposition can be derived for a “vertex” $p^\dagger$ of both $s^\dagger$ and $t^\dagger$. 

**Figure 4.11** Two solids $a$ and $b$ that have overlapping faces with opposite “insides”; their union, intersection and difference. The emphasized area denotes the common face.

**Figure 4.12** Examples to illustrate the validity of the constructed isomorphism for the edges of two solids under the Boolean set operation of sum.
4.2 Solid Modeling

4.2.3 Comparison

Under the algebraic model described in Chapter 2, a shape in $U_{n,k}$ is intrinsically “regular” and so are the shape operations of sum, product, difference and symmetric difference; e.g., given two shapes that share boundary, their product is the empty shape and the difference of one of the shapes with the other is the first shape. The geometric representation for shapes described in Chapter 3 introduces a boundary to a shape. However, this boundary does not constitute a part of the shape, but exists only in order to represent the shape. As such, implicit restrictions do exist on the boundary shape that defines a shape or segment; these follow from Assertion 3.2 and Assertion 3.3. In Chapter 5, we establish a relationship between the boundary of the shape resulting from any of the operations of sum, product, difference and symmetric difference and the boundaries of both operands. This relationship is necessary in order to assert that the closure of $U_{n,k}$ is geometrically also a fact. However, the strength of the shape model and representation does not, primarily, lie in its regularity, as this can fairly easily be guaranteed, not only for shapes, but also for solids under the regularized set operations. First and foremost, its strength lies in its application to shape emergence and subshape detection (see Chapter 9), and the ability to touch any part of a shape. That is, a shape, with a definite description, has indefinitely many parts, any of which can be touched. Therefore, a comparison between shapes and solids and between shape arithmetic and regularized set operations only provides a partial evaluation of the shape model and its representation.

We have seen that an isomorphism exists between $U_{n,k}$ and a subset of $\mathcal{P}(E^k)$ that maps each shape or element in $U_{n,k}$ onto a point set in $E^k$, and that regularized set operations of union, intersection and difference exist that are equivalent to the shape operations of sum, product and difference, under the isomorphism. Such an isomorphism can be used to prove certain properties for shapes, as we have shown in Section 3.1.2 for Assertion 3.1 through Assertion 3.6. In the sequel, we compare the expressive powers of shapes and solids and discuss the uniqueness and rigidity of the maximal representation.

6. In Stouffs and Krishnamurti (1993), a point set formulation is adopted in the proof to the theorem that the algebra $U_{n,k}$ satisfies the axioms of a Boolean ring under symmetric difference and product.
A simple boundary \( a \) in \( U_{n,k} \), under the maximal representation, is represented as a set \( \{ a_1, \ldots, a_m \} \) of maximal segments. We have that a segment \( a_i \) in the representation \( \{ a_1, \ldots, a_m \} \) of a shape \( a \) in \( U_{n,k} \), is a maximal segment if it is disjoint with all other segments in the representation. If \( a_i \) and \( a_j \) \( (i \neq j) \) are not co-equal, then, they are necessarily disjoint, even though their boundaries may overlap, that is, \( B[a_i] \cdot B[a_j] \neq 0 \). Thus, given a shape \( x \) in \( U_{n,k} \), there exist shapes in \( U_{n-2,k} \) that have multiple occurrences within the representational description of \( x \).\(^7\) If there exists a shape in \( U_{n-2,k} \) that has 3 occurrences or more, it follows that the isomorph point set in \( E^k \) is not bounded by an \( (n-1) \)-manifold surface. Therefore, the maximal representation of shapes does not assume shapes to be “bounded by manifold surfaces”. However, the fact that a single shape or segment in \( U_{n-2,k} \) may have multiple occurrences in the representational description of a shape in \( U_{n,k} \), implies that numerical precision errors must be properly dealt with. We refer to Section 8.2 for a discussion regarding arithmetic robustness.

The universe of shapes \( U_0 \times U_1 \times \ldots \times U_n \) is constituted of bounded rectilinear shapes, that is, finite arrangements of spatial elements from among points, line, plane and volume segments, and higher dimensional hyperplane segments, of limited but non-zero measure (this universe may be extended to include curved hyperplane segments). There exists no universal shape; as such, there exists no absolute complement to a shape. Non-regular shapes can be represented using elements from different algebra’s \( U_j \) (see Figure 4.13).

\(^7\) Actually, since a boundary shape in \( U_{n,k} \) corresponds to a closed, \( n \)-dimensional polyhedron in \( E^k \), every boundary segment of this shape has at least 2 occurrences within the representational description.
Such shapes, though non-regular in appearance, are intrinsically regular: Elements belonging to the same algebra compose in a completely regular fashion, while elements belonging to different algebra’s do not compose at all. They exist together, but are handled separately; even though the shapes may be complex in appearance, their representation is simple, and their arithmetic straightforward.

The maximal representation is a canonical representation for shapes. Therefore, any valid representation not only unambiguously models a shape, it also uniquely represents this shape. This trivializes a main problem statement used in solid modeling to evaluate a representation scheme, that is, how to determine whether two solids are equal (Mäntylä, 1988). Solids defined as point sets in $E^3$ have an additional constraint of rigidity. The imposition of this constraint induces the following problem statement: how can it be tested whether a solid can be mapped onto another solid under a rigid transformation (i.e., translation and/or rotation). When applied to shapes, the rigidity test constitutes a subset of the subshape recognition problem, that poses whether a shape is a part of another shape under a Euclidean transformation augmented with scale. Subshape recognition lies at the basis of the shape grammar formalism (see Chapter 9). Both the shape equality relation and subshape recognition take advantage of the canonical representation. It is therefore logical and appropriate to impose such a representation on shapes.
Geometric Modeling
Part II

Arithmetic and Algorithms
Chapter 5
Boundary Evaluation

The arithmetic on shapes has us dealing with the binary operations of sum, product, difference and symmetric difference. The result to each of these operations depends on the geometric relation that exists between the shape operands, that is, whether both shapes are disjoint, overlap, share boundary or if one shape contains the other.

In this chapter we consider the classification approach as a unified approach to each of the operations and relations on shapes and develop the appropriate algorithms. Applied to shape arithmetic, the classification approach consists of classifying the boundary segments of either shape into the classes of inner, outer, same-shared and opposite-shared segments, with respect to the other shape. The main strength of this approach lies in the fact that the classification of the boundary segments of both shapes into the respective classes serves as a unified preprocessing scheme for the consequent determination, in a straightforward and computationally efficient way, of the sum, product, difference and symmetric difference of both shapes, and whether both shapes are disjoint, overlap, share boundary or if one shape contains the other.
Since this approach relies on a classification of the boundary segments of a shape, it is not applicable to shapes of points in $U_0$ (for which the operations and relations are trivial anyway). If we consider points as segments in $U_0$, we can apply the classification approach also to shapes of line segments in $U_1$, even though here too a more straightforward method for determining the results of the operations and relations on shapes exists (fully described in Krishnamurti (1980) and Chase (1989)).

### 5.1 Boundary Evaluation and Merging

Algorithms for determining the regularized set union, intersection, or difference of two solids can be used also to convert solids represented using a different schemata to an equivalent boundary representation. Therefore, algorithms for Boolean operations on boundary representations are sometimes called *boundary evaluation and merging* algorithms (Hoffmann, 1989a). Conceptually, given two solids, the method consists of splitting the faces of both solids along their intersections and of combining certain split faces towards the construction of the resulting solid. This method is complicated by the fact that intersections between faces may occur in a multitude of special cases, such as coincident with an edge or vertex. (Also, the implementation must be robust in the presence of numerical errors.) The *boundary classification* of two solids $A$ and $B$ consists of classifying the split faces of $A$ and $B$ into the sets $A_{in}B$, $A_{out}B$, $B_{in}A$ and $B_{out}A$, respectively, depending on whether the split faces lie inside or outside with respect to the other solid (Mäntylä, 1988). In order to guarantee the regularity of the result, an 8-way *boundary classification* of $A$ and $B$ is considered that adds the sets $A_{on}B^+$, $A_{on}B^-$, $B_{on}A^+$ and $B_{on}A^-$, containing, respectively, the split faces of either solid that coincide with the boundary of the other solid, such that the face normals of the respective faces are either equal (+) or opposite (−). Given this classification, the resulting solid of any of the regularized set operations union, intersection and difference, is constructed from the boundary composition of the appropriate sets, as illustrated in Table 5.1 for a solid $A$ with respect to a solid $B$. 
5.2 Shape Arithmetic

Given that disjoint shapes do not combine under the operations of sum, product, difference and symmetric difference, we consider only co-equal shapes in the sequel. As such, we consider a shorthand notation to denote two co-equal shapes as elements of an algebra $U_n$:

\[ a \in U_n, \ b \in U_n. \]

5.2.1 Boundary Classification

A segment in $U_n$ is deemed inner, outer, same-shared or opposite-shared with respect to a shape in $U_{n+1}$, depending on whether a neighborhood of the segment is contained in the shape. We introduce a classification of the boundary segments of a shape into the classes of same-shared, opposite-shared, inner and outer segments, with respect to another shape.

Let $S^b_a$ denote the class of shared boundary segments, whether same-shared or opposite-shared, of a shape $a$ with respect to a shape $b$.

**Definition 5.1** Given two co-equal shapes $a$ and $b$ in $U_n$, a boundary segment of $a$ is deemed *shared* with respect to $b$, if the segment is a part of the boundary of $b$;

\[ \forall a, b \in U_n, \forall s \leq B[a]: s \leq S^b_a \iff s \leq B[b]. \]

**Property**

\[ S^b_a = B[a] \cdot B[b] = S^a_b \]  \[\text{[eq5.1]}\]

**Proof**: Follows directly from the above definition.

---

<table>
<thead>
<tr>
<th></th>
<th>AinB</th>
<th>AoutB</th>
<th>BinA</th>
<th>BoutA</th>
<th>AonB+</th>
<th>AonB−</th>
</tr>
</thead>
<tbody>
<tr>
<td>Union, $A \cup B$</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Intersection, $A \cap B$</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Difference, $A \setminus B$</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 5.1** Set-membership classification for the Boolean set operations of union, intersection and difference into the 8-way boundary classification sets (here only for $A$ with respect to $B$).
Figure 5.1 illustrates the shared boundary segments for two shapes in $U_3$. Let $M^b_a$ denote the class of same-shared segments of a shape $a$ with respect to a shape $b$.

**Definition 5.2** Given two co-equal shapes $a$ and $b$ in $U_n$, a boundary segment $s$ of $a$ is deemed *same-shared* with respect to $b$, if $s$ is deemed shared with respect to $b$ and there exists a neighborhood of $s$ that is a part of both $a$ and $b$;

$$\forall a, b \in co_{U_n}, \forall s \leq B[a]; s \leq M^b_a \iff s \leq S^b_a \wedge (\exists \Delta(s) \in U_n; \Delta(s) \leq a \wedge \Delta(s) \leq b).$$

It follows directly from the definition and [eq5.1] that $M^b_a = M^a_b$. Let $N^b_a$ denote the class of opposite-shared segments of a shape $a$ with respect to a shape $b$.

**Definition 5.3** Given two co-equal shapes $a$ and $b$ in $U_n$, a boundary segment $s$ of $a$ is deemed *opposite-shared* with respect to $b$, if $s$ is deemed shared with respect to $b$ and there exists a neighborhood of $s$ that is a part of $a$ such that the product of this neighborhood with $b$ equals $0$;

$$\forall a, b \in co_{U_n}, \forall s \leq B[a]; s \leq N^b_a \iff s \leq S^b_a \wedge (\exists \Delta(s) \in U_n; \Delta(s) \leq a \wedge \Delta(s) \cdot b = 0).$$

Figure 5.2 illustrates the same-shared and opposite-shared boundary segments for two shapes in $U_3$.

The following identities hold for all $a, b \in U_n$:

$$M^b_a = S^b_a \cdot B[a \cdot b] \quad \text{[eq5.2]}$$

$$N^b_a = S^b_a \cdot B[a - b] \quad \text{[eq5.3]}$$

**Proof:** [eq5.2]: Take any segment $s$ that is a part of $S^b_a \cdot B[a \cdot b]$. Then $s \leq S^b_a$ and $s$ is a boundary segment of $a \cdot b$. Therefore, $a \cdot b$ constitutes a neighborhood of $s$ and $a \cdot b \leq a$ and
Thus, using the definition of $M^b_a \cdot s \leq M^b_a$.

Conversely, given $s \leq M^b_a$, we have that $\exists \Delta_1(s) \in \mathcal{U}_a$; $\Delta_1(s) \leq a$ and $s \leq B[a]$. From $s \leq B[a]$, it follows that $\exists \Delta_2(s) \in \mathcal{U}_a$; $\Delta_2(s) \cdot a = 0$, or $\Delta_2(s) \cdot (a - b) = 0$. It follows from Assertion 3.2 that $s$ is a boundary segment of $a \cdot b$.

[eq5.3]: Take any segment $s$ that is a part of $S^b_a \cdot B[a - b]$. Then $s \leq S^b_a$ and $s$ is a boundary segment of $a - b$. Therefore, $a - b$ is a neighborhood of $s$ and $a - b \leq a$ and $(a - b) \cdot b = 0$. Thus, from the definition of $N^b_a$, $s \leq N^b_a$.

Conversely, given $s \leq N^b_a$, we have that $\exists \Delta_1(s) \in \mathcal{U}_a$; $\Delta_1(s) \leq a - b$ and $s \leq B[a]$. From $s \leq B[a]$, it follows that $\exists \Delta_2(s) \in \mathcal{U}_a$; $\Delta_2(s) \cdot a = 0$, or $\Delta_2(s) \cdot (a - b) = 0$. It follows from Assertion 3.2 that $s$ is a boundary segment of $a - b$.

Lemma 5.1 Given two co-equal shapes in $\mathcal{U}_a$, taking one with respect to the other, the classes of same-shared and opposite-shared boundary segments form a partitioning of the class of shared segments;

$$\forall a, b \in \mathcal{U}_a: \quad S^b_a = M^b_a + N^b_a \wedge M^b_a \cdot N^b_a = 0.$$  

Proof: Using [eq5.2] and [eq5.3], we have that $M^b_a + N^b_a = (S^b_a \cdot B[a \cdot b]) + (S^b_a \cdot B[a - b]) = S^b_a \cdot (B[a \cdot b] + B[a - b]) = S^b_a$, if and only if $S^b_a \leq B[a \cdot b] + B[a - b]$.

Given $(a \cdot b) + (a - b) = a$, we derive (using Assertion 3.3) that $B[a] \leq B[a \cdot b] + B[a - b]$ and, thus, $S^b_a \leq B[a] \leq B[a \cdot b] + B[a - b]$.

Assume $M^b_a \cdot N^b_a \neq 0$. Then, $\exists s \in M^b_a \cdot N^b_a$, $s \neq 0$.

From $s \in M^b_a$, it follows that $\exists \Delta_1(s) \in \mathcal{U}_a$; $\Delta_1(s) \leq a \wedge \Delta_1(s) \leq b$, and from $s \in N^b_a$ it follows
that $\exists \Delta_2(s) \in U_n; \Delta_2(s) \leq a \land \Delta_2(s) \cdot b = 0$. Given $\Delta_1(s) \leq b$ and $\Delta_2(s) \cdot b = 0$ it follows that $\Delta_1(s) \cdot \Delta_2(s) = 0$. Since $\Delta_1(s) \leq a \land \Delta_2(s) \leq a$, it is impossible for $s$ to be a boundary segment of $a$ (Assertion 3.1), and this violates the assumption that $s \in M_a \cdot N_a$.

Thus, $M_a \cdot N_a = 0$.  

Lemma 5.2 Given two co-equal shapes in $U_n$ the class of opposite-shared boundary segments is the same for either shape with respect to the other;

$\forall a, b \in co U_n$:  

$N_a^b = N_a^b$.  

Proof: Follows directly from Lemma 5.1 and the facts that $S_a^b = S_b^a$ and $M_a^b = M_b^a$.  

We can, therefore, use $N$ to denote $N_a^b$ (or $N_b^a$) if it is obvious from the context which shapes $a$ and $b$ are meant and, likewise, use $M$ to denote $M_a^b$ (or $M_b^a$) and $S$ to denote $S_a^b$ (or $S_b^a$).

Let $I_a$ denote the class of inner segments and $O_a$ denote the class of outer segments, of a shape $a$ with respect to a shape $b$.

Definition 5.4 Given two co-equal shapes $a$ and $b$ in $U_n$, a boundary segment $s$ of $a$ is deemed inner with respect to $b$, if no part of $s$ is deemed shared with respect to $b$ and there exists a neighborhood of $s$ that is a part of $b$;

$\forall a, b \in co U_n, \forall s \leq B[a]: s \leq I_a^b \iff s \cdot S = 0 \land (\exists \Delta(s) \in U_n; \Delta(s) \leq b)$.  

Definition 5.5 Given two co-equal shapes $a$ and $b$ in $U_n$, a boundary segment $s$ of $a$ is deemed outer with respect to $b$, if no part of $s$ is deemed shared with respect to $b$ and there exists a neighborhood of $s$ such that the product of this neighborhood with $b$ equals 0;

$\forall a, b \in co U_n, \forall s \leq B[a]: s \leq O_a^b \iff s \cdot S = 0 \land (\exists \Delta(s) \in U_n; \Delta(s) \cdot b = 0)$.  

We use $I_a$ to denote $I_a^b$ and $O_a$ to denote $O_a^b$ if it is obvious from the context which shape $b$ is meant. Figure 5.3 illustrates the inner boundary segments of two shapes in $U_3$; Figure 5.4 illustrates the outer boundary segments.

The following identities hold for all $a, b \in co U_n$;

$s \leq I_a \Rightarrow \exists \Delta(s) \in U_n; \Delta(s) \leq a \cdot b$  

$[eq5.4]$  

$s \leq O_a \Rightarrow \exists \Delta(s) \in U_n; \Delta(s) \leq a - b$  

$[eq5.5]$.  

We use $I_a$ to denote $I_a^b$ and $O_a$ to denote $O_a^b$ if it is obvious from the context which shape $b$ is meant. Figure 5.3 illustrates the inner boundary segments of two shapes in $U_3$; Figure 5.4 illustrates the outer boundary segments.
5.2 Shape Arithmetic

\[ I_a = (B[a \cdot b] \cdot B[a]) \cdot S = B[a \cdot b] - B[b] \]  

[eq5.6]

\[ O_a = (B[a - b] \cdot B[a]) \cdot S = B[a - b] - B[b] \]  

[eq5.7]

**Proof**: [eq5.4]: Take any segment \( s \leq B[a] \). We have that \( \exists \Delta(s) \in U_n; \Delta(s) \leq a \). Using Assertion 3.3, we define a partitioning on \( s \) and \( \Delta(s) \) with respect to \( b \) as follows:

\[ s = t + (s - t), \Delta(t) = \Delta(s) \cdot b \text{ and } \Delta(s - t) = \Delta(s) - b. \]

For \( t \leq I_a \), we have that \( \Delta(t) \leq \Delta(s) \leq a \text{ and } \Delta(t) \leq b \).

For \( s - t \leq I_a \), from the definition of \( I_a \), we have that there exists a neighborhood \( \Delta'(s - t) \in U_n \) such that \( \Delta'(s - t) \leq b \). But, \( \Delta(s - t) = \Delta(s) - b \) implies that \( \Delta(s - t) \cdot b = 0 \). Therefore, \( s - t \) must be a part of the boundary of \( b \), which contradicts the fact that \( s - t \leq I_a \). Thus, \( t = s \) and \( \Delta(s) \leq a \cdot b \).

[eq5.5]: Take any segment \( s \leq B[a] \). We have that \( \exists \Delta(s) \in U_n; \Delta(s) \leq a \). Using Assertion 3.3, we define a partitioning on \( s \) and \( \Delta(s) \) with respect to \( b \) as follows:

**Figure 5.3** The boundary segments of either of two shapes \( a \) and \( b \) that are deemed inner with respect to the other shape.

**Figure 5.4** The boundary segments of either of two shapes \( a \) and \( b \) that are deemed outer with respect to the other shape.
\( s = t + (s - t), \Delta(t) = \Delta(s) \cdot b \) and \( \Delta(s - t) = \Delta(s) - b. \)

For \( s - t \leq O_a \), we have that \( \Delta(s - t) \leq \Delta(s) \leq a \) and \( \Delta(s - t) \cdot b = 0. \)

For \( t \leq O_a \), from the definition of \( O_a \), we have that there exists a neighborhood \( \Delta'(t) \in U_n \) such that \( \Delta'(t) \cdot b = 0. \) But, \( \Delta(t) \leq b. \) Thus, \( t \) must be a part of the boundary of \( b \), which contradicts the fact that \( t \leq I_a. \) Thus, \( t = 0 \) and \( \Delta(s) \cdot b = 0 \) as well as \( \Delta(s) \leq a. \)

[eq5.6]: Take any segment \( s \) that is a part of \((B[a \cdot b] \cdot B[a]) - S\). Then, \( s \cdot S = 0 \) and \( s \) is a boundary segment of \( a \cdot b \) and a boundary segment of \( a \). Given that \( s \cdot B[b] = 0 \) and \( a \cdot b \leq b \), [eq3.1] specifies that there exists a neighborhood \( \Delta(s) \) of \( s \) such that \( \Delta(s) \leq b \), or \( s \leq I_a. \)

Conversely, from \( s \leq I_a \) it follows that \( s \leq B[a], s \cdot B[b] = 0 \) and \( \nexists \Delta_1(s) \in U_n: \Delta_1(s) \leq a \cdot b \) (ieq5.4). From \( s \leq B[a] \) it follows that \( \nexists \Delta_2(s) \in U_n: \Delta_2(s) \cdot a = 0 \) and, therefore, \( \Delta_2(s) \cdot (a \cdot b) = 0. \) Then, Assertion 3.2 specifies that \( s \leq B[a \cdot b], \) or \( s \leq B[a \cdot b] \cdot B[a]. \) Since \( s \cdot S = 0, \) we have that \( s \leq (B[a \cdot b] \cdot B[a]) - S. \)

Also, using [eq5.1], [eq2.20] and Theorem 3.4, we have that \((B[a \cdot b] \cdot B[a]) - S = (B[a \cdot b] \cdot B[a]) - (B[b] \cdot B[a]) = (B[a \cdot b] - B[b]) \cdot B[a] = B[a \cdot b] - B[b].\)

[eq5.7]: Take any segment \( s \) that is a part of \((B[a - b] \cdot B[a]) - S\). Then \( s \cdot S = 0 \) and \( s \) is a boundary segment of \( a - b \) and a boundary segment of \( a \). Thus, \( a - b \) is a neighborhood of \( s \) and \( (a - b) \cdot b = 0 \), or \( s \leq O_a. \)

Conversely, from \( s \leq O_a \) it follows that \( s \leq B[a], s \cdot B[b] = 0 \) and \( \nexists \Delta_2(s) \in U_n: \Delta_2(s) \cdot a = 0 \) and, therefore, \( \Delta_2(s) \cdot (a - b) = 0. \) Then, Assertion 3.2 specifies that \( s \leq B[a \cdot b], \) or \( s \leq B[a \cdot b] \cdot B[a]. \) Since \( s \cdot S = 0, \) we have that \( s \leq (B[a - b] \cdot B[a]) - S. \)

Also, using [eq5.1], [eq2.20] and Theorem 3.4, we have that \((B[a - b] \cdot B[a]) - S = (B[a - b] \cdot B[a]) - (B[b] \cdot B[a]) = (B[a - b] - B[b]) \cdot B[a] = B[a - b] - B[b].\)

\[ \square \]

**Lemma 5.3** Given two co-equal shapes in \( U_n \), taking one with respect to the other, the classes of inner, outer and shared boundary segments form a partitioning of the boundary of the shape;

\[ \forall a, b \in_{co} U_n: I_a + O_a + S = B[a] \land I_a \cdot O_a = 0 \land I_a \cdot S = 0 \land S \cdot O_a = 0. \]
5.2 Shape Arithmetic

Proof: We first prove that \( I_a + O_a + S = B[a] \).

\[
I_a + O_a + S = ((B[a \cdot b] \cdot B[a]) - S) + ((B[a - b] \cdot B[a]) - S) + S = (B[a \cdot b] \cdot B[a]) + (B[a - b] \cdot B[a]) + S = ((B[a \cdot b] + B[a - b]) \cdot B[a]) + (B[a] \cdot B[b]) = (B[a \cdot b] + B[a - b] + B[b]) \cdot B[a].
\]

Using Theorem 3.4, we derive that \( B[a \cdot b] + B[a - b] + B[b] \leq B[a] + B[b] \).

Conversely, since \( B[a] \leq B[a \cdot b] + B[a - b] \) (from \((a \cdot b) + (a - b) = a \) and Theorem 3.4), we have that \( B[a] + B[b] \leq B[a \cdot b] + B[a - b] + B[b] \). Thus, \( B[a \cdot b] + B[a - b] + B[b] = B[a] + B[b] \) and \( I_a + O_a + S = (B[a] + B[b]) \cdot B[a] = B[a] \).

\( I_a \cdot S = 0 \) and \( S \cdot O_a = 0 \) follow directly from the definitions of \( I_a \) and \( O_a \).

Assume \( I_a \cdot O_a \neq 0 \). Then, \( \exists s \in I_a \cdot O_a, s \neq 0 \). From \( s \in I_a \) it follows that \( \exists \Delta_1(s) \in U_n; \Delta_1(s) \leq b \), and from \( s \in O_a \) it follows that \( \exists \Delta_2(s) \in U_n; \Delta_2(s) \cdot b = 0 \). Given \( \Delta_1(s) \leq b \) and \( \Delta_2(s) \cdot b = 0 \) it follows that \( s \) is a boundary segment of \( b \). However, \( s \) is a boundary segment of \( a \) and, thus, \( s \) is a part of \( S \). Since \( I_a \cdot S = 0 \) and \( S \cdot O_a = 0 \), we conclude that \( I_a \cdot O_a = 0 \).

5.2.2 Shape Operations

The boundary of the shape resulting from any of the binary shape operations equals the sum of the shapes from the appropriate classes. The following lemmas state which classes constitute part of the resulting boundary shape for each of the operations; this information is summarized in Table 5.2.1.

Consider a shape \( a \) in \( U_n \), the boundary segments of which are classified with respect to another shape \( b \), also in \( U_n \), with \( a \) and \( b \) co-equal. Take any boundary segment \( l \) of \( a \) and let \( \text{left}(l) \) denote a neighborhood of \( l \) that is a part of \( a \), such that \( \text{left}(l) \) is either a part of \( b \) or the nearby objects. These results are given also in Stouffs and Krishnamurti (1992), where they are proven using a point set formulation. Here we use a more direct proof that does not rely, explicitly, on the isomorphism between shapes and point sets but uses the notion of neighborhoods as established in Chapter 3.
product of \( l \) with \( b \) equals 0. From the above definitions, [eq5.4] and [eq5.5], such a neighborhood must exist if \( l \) is deemed wholely inner, outer, same-shared or opposite-shared with respect to \( b \). Similarly, let \( \text{right}(l) \) denote a neighborhood of \( l \) such that \( \text{right}(l) \cdot a = 0 \) and either \( \text{right}(l) \leq b \) or \( \text{right}(l) \cdot b = 0 \). Again, such a neighborhood exists if \( l \) is deemed wholly inner, outer, same-shared or opposite-shared with respect to \( b \). The proofs of the lemmas below rely on these constructions of the neighborhoods \( \text{left}(l) \) and \( \text{right}(l) \) for a boundary segment \( l \) of a shape \( a \) classified with respect to a shape \( b \).

<table>
<thead>
<tr>
<th></th>
<th>Sum</th>
<th>Product</th>
<th>Difference</th>
<th>Symmetric difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( a + b )</td>
<td>( a \cdot b )</td>
<td>( a - b )</td>
<td>( a \oplus b )</td>
</tr>
<tr>
<td>Inner ((a/b))</td>
<td>- / -</td>
<td>✔/ ✔</td>
<td>- / ✔</td>
<td>✔/ ✔</td>
</tr>
<tr>
<td>Outer ((a/b))</td>
<td>✔/ ✔</td>
<td>- / -</td>
<td>✔/ -</td>
<td>✔/ ✔</td>
</tr>
<tr>
<td>Same-shared</td>
<td>✔</td>
<td>✔</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Opposite-shared</td>
<td>-</td>
<td>-</td>
<td>✔</td>
<td>-</td>
</tr>
</tbody>
</table>

**Table 5.2** Set-membership classification for the operations of sum, product, difference and symmetric difference, applied to two shapes \( a \) and \( b \). The participation of a class into the solution set is indicated by the symbol ✔. The symbol “-” indicates that a class does not participate into the solution set.

**Lemma 5.4** The boundary of the sum of two co-equal shapes equals the sum of the boundary segments from either shape that are deemed outer with respect to the other shape, and of the same-shared boundary segments:

\[
\forall a, b \in co \cup n: B[a + b] = O_a + O_b + M.
\]

**Proof:** Let \( l \) denote an arbitrary boundary segment of \( a \) such that it is deemed wholly inner, outer, same-shared or opposite-shared, with respect to \( b \). Let \( \text{left}(l) \) and \( \text{right}(l) \) denote neighborhoods of \( l \) as constructed above. Since \( \text{left}(l) \) is a part of \( a \) it is also a part of \( a + b \). Thus \( l \) is a boundary segment of \( a + b \) if and only if \( \text{right}(l) \) is not a part of \( a + b \), and
therefore if right\( (l) \) \( \cdot b = 0 \). This is the case if \( l \) is deemed outer or same-shared with respect to \( b \) (as illustrated in Figure 5.5 for \( U_2 \)). The proof is similar for an arbitrary boundary segment of \( b \) with respect to \( a \).

**Lemma 5.5** The boundary of the product of two co-equal shapes equals the sum of the boundary segments from either shape that are deemed inner with respect to the other shape, and of the same-shared boundary segments;

\[
\forall a, b \in \text{co } U_n:\quad B[a \cdot b] = I_a + I_b + M.
\]

**Proof:** Let \( l \) denote an arbitrary boundary segment of \( a \) such that it is deemed wholly inner, outer, same-shared or opposite-shared, with respect to \( b \). Let left\( (l) \) and right\( (l) \) denote neighborhoods of \( l \) as constructed above. Since right\( (l) \cdot a = 0 \) it follows that right\( (l) \cdot (a \cdot b) = 0 \). Thus \( l \) is a boundary segment of \( a \cdot b \) if and only if left\( (l) \) is a part of \( a \cdot b \), and therefore if left\( (l) \leq b = 0 \). This is the case if \( l \) is deemed inner or same-shared with respect to \( b \). This is illustrated in Figure 5.6 for \( U_2 \), which is derived from Figure 5.5 by replacing \( \left[ l \text{ outer } \Rightarrow \left[ \text{left}(l) \cdot b = 0 \right] \right] \) by \( \left[ l \text{ inner } \Rightarrow \left[ \text{left}(l) \leq b \right] \right] \). The proof is similar for an arbitrary boundary segment of \( b \) with respect to \( a \).
Remark: There exists a more direct proof using \[eq5.2\], \[eq5.6\] and \[eq5.7\]:

\[ I_a + I_b + M = ((B[a \cdot b] \cdot B[a]) - S) + ((B[a \cdot b] \cdot B[b]) - S) + (S \cdot B[a \cdot b]) \]
\[ = ((B[a \cdot b] \cdot B[a]) + (B[a \cdot b] \cdot B[b])) - S + (B[a \cdot b] \cdot S) \]
\[ = ((B[a \cdot b] \cdot (B[a] + B[b])) - S) + (B[a \cdot b] \cdot S) \]
\[ = (B[a \cdot b] - S) + (B[a \cdot b] \cdot S) = B[a \cdot b]. \]

Figure 5.6 Inner and outer neighborhoods of a boundary segment \( l \) of \( a \) with respect to shapes \( a \) and \( b \). Only if \( l \) is deemed inner or same-shared with respect to \( b \), then \( l \) is also a boundary segment of \( a \cdot b \).

Lemma 5.6 The boundary of the difference of one shape with respect to another, co-equal shape, equals the sum of the boundary segments from the first shape that are deemed outer, and of the boundary segments from the second shape that are deemed inner, with respect to the other shape, as well as of the opposite-shared boundary segments:

\[ \forall a, b \in co, U_a : \quad B[a - b] = O_a + I_b + N. \]

Proof: Let \( l \) denote an arbitrary boundary segment of \( a \) such that it is deemed wholly inner, outer, same-shared or opposite-shared, with respect to \( b \). Let \( left(l) \) and \( right(l) \) denote neighborhoods of \( l \) as constructed above. Since \( right(l) \cdot a = 0 \) it follows that \( right(l) \cdot (a - b) = 0 \). Thus \( l \) is a boundary segment of \( a - b \) if and only if \( left(l) \) is a part of \( a - b \), and therefore if \( left(l) \cdot b = 0 \). This is the case if \( l \) is deemed outer or opposite-shared.
with respect to \( b \). An illustration can be conceived similar to Figure 5.5 with “\( \text{\textit{l same-shared}} \Rightarrow \text{\textit{left(l)}} \leq b \ \backslash \ \text{\textit{right(l)}} \cdot b = 0 \)" replaced by “\( \text{\textit{l opposite-shared}} \Rightarrow \text{\textit{left(l)}} \cdot b = 0 \ \backslash \ \text{\textit{right(l)}} \leq b \)”. The proof is similar, but opposite, for an arbitrary boundary segment of \( b \) with respect to \( a \).

**Lemma 5.7** The boundary of the symmetric difference of two co-equal shapes equals the sum of all inner and outer boundary segments from either shape:

\[
\forall a, b \in \text{co} U_n : B[a \oplus b] = I_a + I_b + O_a + O_b.
\]

**Proof:** Let \( l \) denote an arbitrary boundary segment of \( a \) such that it is deemed wholly inner, outer, same-shared or opposite-shared, with respect to \( b \). Let \textit{left}(\( l \)) and \textit{right}(\( l \)) denote neighborhoods of \( l \) as constructed above. Since \textit{left}(\( l \)) is a part of \( a \), if \textit{left}(\( l \)) \cdot b = 0 \) it follows that \textit{left}(\( l \)) is a part of \( a \oplus b \). Since \textit{right}(\( l \)) \cdot a = 0 \), only if \textit{right}(\( l \)) \cdot b = 0 \) is \textit{right}(\( l \)) \cdot (a \oplus b) = 0 \). In this case, \( l \) is deemed outer with respect to \( b \). Similarly if both \textit{left}(\( l \)) and \textit{right}(\( l \)) are parts of \( b \), and therefore \( l \) is deemed inner with respect to \( b \), then \( l \) is a subset of the boundary of \( a \oplus b \). An illustration similar to Figure 5.5, can be conceived with “\( \text{\textit{l same-shared}} \Rightarrow \text{\textit{left(l)}} \leq b \ \backslash \ \text{\textit{right(l)}} \cdot b = 0 \)" replaced by “\( \text{\textit{l inner}} \Rightarrow \text{\textit{left(l)}} \leq b \ \backslash \ \text{\textit{right(l)}} \leq b \)”. The proof is similar for an arbitrary boundary segment of \( b \) with respect to \( a \).

Each of the four lemmas above translates into a simple procedure\(^2\), if we assume the existence of two other procedures: The procedure \textsc{classify} determines the classes of boundary segments of each shape with respect to the other shape (see Algorithm 6.4 for shapes in \( U_2 \) and Algorithm 6.7 for shapes in \( U_3 \), in Chapter 6). The procedure \textsc{construct} determines the shape defined by a given boundary shape (see Algorithm 7.2 for shapes in \( U_2 \) and Algorithm 7.5 for shapes in \( U_3 \), in Chapter 7). Each of the procedures takes as input two co-equal shapes in the same algebra and returns the shape resulting from the respective operation, when applied to the two input shapes.

---

\(^2\) The pseudo-code form and conventions are due to Cormen et al. (1990). The symbol \( \cup \) denotes the merging of (sorted) sets of line segments.
Algorithm 5.1

\begin{algorithm}
\textbf{SUM} (a, b) \\
1 \quad (I_a, I_b, M, N, O_a, O_b) \leftarrow \text{CLASSIFY} (a, b) \\
2 \quad \text{return CONSTRUCT} (O_a \cup O_b \cup M)
\end{algorithm}

\begin{algorithm}
\textbf{PRODUCT} (a, b) \\
1 \quad (I_a, I_b, M, N, O_a, O_b) \leftarrow \text{CLASSIFY} (a, b) \\
2 \quad \text{return CONSTRUCT} (I_a \cup I_b \cup M)
\end{algorithm}

\begin{algorithm}
\textbf{DIFFERENCE} (a, b) \\
1 \quad (I_a, I_b, M, N, O_a, O_b) \leftarrow \text{CLASSIFY} (a, b) \\
2 \quad \text{return CONSTRUCT} (O_a \cup I_b \cup N)
\end{algorithm}

\begin{algorithm}
\textbf{SYMMETRIC-DIFFERENCE} (a, b) \\
1 \quad (I_a, I_b, M, N, O_a, O_b) \leftarrow \text{CLASSIFY} (a, b) \\
2 \quad \text{return CONSTRUCT} (I_a \cup I_b \cup O_a \cup O_b)
\end{algorithm}

5.2.3 Shape Relations

The relation between two co-equal shapes depends on the distribution of the boundary segments of each shape into the four classes: In particular, it depends on whether or not each of the classes contains one or more segments. The particular dependencies are stated in the following lemmas and are summarized in Table 5.3 and Table 5.4.

Lemma 5.8 A shape a contains a, co-equal, shape b if and only if there exists no boundary segment of a that is deemed inner with respect to b, and no such segment of b that is deemed outer with respect to a, and there exists no opposite-shared boundary segment of a and b:

\[ \forall a, b \in \text{co } U_a : \quad b \leq a \iff I_a = 0 \land O_b = 0 \land N = 0. \]

Proof: A shape a contains a shape b if and only if b is a part of a and, therefore, if and only if the difference of b with respect to a equals 0. From Lemma 5.6, the boundary shape of this difference consists of the boundary segments from a that are deemed inner with respect to b,
the boundary segments from \( b \) that are deemed outer with respect to \( a \), and the opposite-shared boundary segments. Thus \( b - a = 0 \) if and only if all three classes equal 0.

\[ b \leq a \quad a \cdot b = 0 \]
\[ B[a] \cdot B[b] \neq 0 \quad B[a] \cdot B[b] = 0 \]

<table>
<thead>
<tr>
<th></th>
<th>Contain ( b \leq a )</th>
<th>Share boundary ( a \cdot b = 0 )</th>
<th>Disjoint ( B[a] \cdot B[b] \neq 0 )</th>
<th>Disjoint ( B[a] \cdot B[b] = 0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inner ((a/b))</td>
<td>✔/ -</td>
<td>✔/ ✔</td>
<td>✔/ ✔</td>
<td>✔/ ✔</td>
</tr>
<tr>
<td>Outer ((a/b))</td>
<td>✔/ -</td>
<td>✔/ ✔</td>
<td>✔/ ✔</td>
<td>✔/ ✔</td>
</tr>
<tr>
<td>Same-shared</td>
<td>-</td>
<td>-</td>
<td>✔/ ✔</td>
<td>✔/ ✔</td>
</tr>
<tr>
<td>Opposite-shared</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

Table 5.3  Set-membership conditions for the relations contain, share boundary and disjoint, applied to two plane segments \( a \) and \( b \). Conditions are denoted by either the symbol ✔ or ✗, respectively indicating whether the class imperatively or impermissibly contains one or more segments. The symbol “-” denotes indifference.

<table>
<thead>
<tr>
<th></th>
<th>i</th>
<th>ii</th>
<th>iii</th>
<th>iv</th>
<th>v</th>
<th>vi</th>
<th>vii</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inner ((a/b))</td>
<td>✔/ -</td>
<td>-/✔</td>
<td>✔/✔</td>
<td>✔/ -</td>
<td>✔/ -</td>
<td>-/✔</td>
<td>-/ -</td>
</tr>
<tr>
<td>Outer ((a/b))</td>
<td>✔/ -</td>
<td>-/✔</td>
<td>-/ -</td>
<td>✔/✔</td>
<td>-/ -</td>
<td>-/ -</td>
<td>-/ -</td>
</tr>
<tr>
<td>Same-shared</td>
<td>-</td>
<td>-</td>
<td>✔</td>
<td>-</td>
<td>-</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Opposite-shared</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>✗</td>
<td>✔</td>
<td>✗</td>
<td>✔</td>
</tr>
</tbody>
</table>

Table 5.4  Set-membership condition sets for the relation overlap, applied to two plane segments \( a \) and \( b \). A condition is denoted by the symbol ✔, indicating that the respective class needs to be non-zero. Each set defines a sufficient condition for \( a \) and \( b \) to overlap.
Lemma 5.9 Two co-equal shapes \(a\) and \(b\) overlap (without containment) if and only if
\[
(I_a \neq 0 \land O_a \neq 0) \lor (I_b \neq 0 \land O_b \neq 0) \lor (I_a \neq 0 \land I_b \neq 0) \\
(O_a \neq 0 \land O_b \neq 0 \land M \neq 0) \lor (I_a \neq 0 \land N \neq 0) \lor (I_b \neq 0 \land N \neq 0) \lor (M \neq 0 \land N \neq 0)
\]

Proof: Two shapes \(a\) and \(b\) overlap (possibly with containment) if and only if the product of \(a\) and \(b\) is non-zero. If the differences of either shape with respect to the other shape are non-zero, then \(a\) does not contain \(b\) and neither does \(b\) contain \(a\). Thus \(a\) and \(b\) overlap, without containment, if and only if \((a \cdot b \neq 0) \land (a - b \neq 0) \land (b - a \neq 0)\).

Lemma 5.5 and Lemma 5.6 specify that \(B[a \cdot b] = I_a + I_b + M, B[a - b] = O_a + I_b + N\) and \(B[b - a] = I_a + O_b + N\), or, conversely, \(a \cdot b = \Gamma(I_a + I_b + M), a - b = \Gamma(O_a + I_b + N)\) and \(b - a = \Gamma(I_a + O_b + N)\). It follows that

\[
(a \cdot b \neq 0) \land (a - b \neq 0) \land (b - a \neq 0)
\]
\[
\Leftrightarrow (I_a + I_b + M \neq 0) \land (O_a + I_b + N \neq 0) \land (I_a + O_b + N \neq 0)
\]
\[
\Leftrightarrow (I_a \neq 0 \lor I_b \neq 0 \lor M \neq 0) \land (O_a \neq 0 \lor I_b \neq 0 \lor N \neq 0) \land (I_a \neq 0 \lor O_b \neq 0 \lor N \neq 0)
\]
\[
\Leftrightarrow (I_a \neq 0 \lor O_a \neq 0) \lor (I_a \neq \emptyset \land I_b \neq 0) \lor (I_a \neq 0 \land N \neq 0) \lor (I_b \neq 0 \land N \neq 0) \lor [M \neq 0 \land [(O_a \neq 0 \land I_a \neq 0) \lor (O_a \neq 0 \land O_b \neq 0) \lor (O_b \neq \emptyset \land I_a \neq 0) \lor (I_b \neq 0 \land O_b \neq 0)]
\]
\[
\Leftrightarrow (I_a \neq 0 \land O_a \neq 0) \lor (I_a \neq \emptyset \land I_b \neq 0) \lor (I_a \neq 0 \land N \neq 0) \lor (I_b \neq 0 \land O_b \neq 0) \lor (I_b \neq 0 \land N \neq 0) \lor (O_a \neq 0 \land O_b \neq 0 \land M \neq 0) \lor (M \neq 0 \land N \neq 0)
\]

Figure 5.7 illustrates graphically that each of the conjunctions (i through vii) is necessary as a (sufficient) condition for two shapes in \(U_2\) to overlap.

Lemma 5.10 Two co-equal shapes \(a\) and \(b\) share boundary (without containment or overlap) if and only if there exists no boundary segment of \(a\) or \(b\) that is deemed inner with respect to the other, and there exists no same-shared boundary segment of \(a\) and
5.2 Shape Arithmetic

**Proof**: Two shapes $a$ and $b$ do not overlap (neither does one contain the other) if and only if the product of $a$ and $b$ equals 0. From Lemma 5.5, the boundary shape of this product
consists of the boundary segments from either shape that are deemed inner with respect to the other shape, and of the same-shared boundary segments. Thus, $a \cdot b = 0$ if and only if all three classes equal 0. In order for $a$ and $b$ to share boundary, the class of opposite-shared boundary segments needs to be non-zero.

**Lemma 5.11** Two co-equal shapes $a$ and $b$ are disjoint (without sharing boundary) if and only if there exists no boundary segment of $a$ or $b$ that is deemed inner with respect to the other shape, and there exists neither a same-shared boundary segment nor an opposite-shared boundary segment of $a$ and $b$:

$$\forall a, b \in_{co} U_n : a \cdot b = 0 \land B[a] \cdot B[b] = 0 \iff I_a = 0 \land I_b = 0 \land M = 0 \land N = 0.$$

**Proof:** Two shapes $a$ and $b$ are disjoint (without sharing boundary) if and only if the product of $a$ and $b$ equals 0, and if $a$ and $b$ do not share boundary segments. Thus, the proof follows from Lemma 5.10.

It follows naturally from Lemma 5.11 that, if shapes $a$ and $b$ are disjoint, all boundary segments of both $a$ and $b$ are deemed outer with respect to the other shape. The procedures in Algorithm 5.2 implement the results of the above lemmas, for co-equal shapes.

Lemma 5.8 serves as a constructive definition for the embedding of co-equal shapes in $U_n$ ($n \geq 2$). A shape $a$ contains a shape $b$ if $b$ is a part of $a$; a part is an embedding of one shape in another shape (Definition 2.1). Definition 3.14 specifies the embedding of a shape in $U_n$ in terms of the embedding of (maximal) segments, also in $U_{n'}$. Then, Lemma 5.8 relates the embedding of shapes in $U_n$ to (classes) of shapes in $U_{n-1}$. The construction of these classes is described, algorithmically, in Section 6.2 for classes of segments in $U_1$ and in Section 6.3 for classes of segments in $U_2$, the latter serving to define the embedding of shapes in $U_3$. 
Algorithm 5.2

**CONTAIN** \((a, b)\)
1. \((I_a, I_b, M, N, O_a, O_b) \leftarrow \text{CLASSIFY} (a, b)\)
2. if \(I_a = 0\) and \(O_b = 0\) and \(N = 0\)
   3. then return \(\text{TRUE}\)
   4. else return \(\text{FALSE}\)

**OVERLAP** \((a, b)\)
1. \((I_a, I_b, M, N, O_a, O_b) \leftarrow \text{CLASSIFY} (a, b)\)
2. if \((I_a \neq 0 \text{ and } O_a \neq 0)\) or \((I_b \neq 0 \text{ and } O_b \neq 0)\) or \((I_a \neq 0 \text{ and } I_b \neq 0)\) or \((O_a \neq 0 \text{ and } O_b \neq 0 \text{ and } M \neq 0)\) or \((I_a \neq 0 \text{ and } N \neq 0)\) or \((I_b \neq 0 \text{ and } N \neq 0)\) or \((M \neq 0 \text{ and } N \neq 0)\)
   3. then return \(\text{TRUE}\)
   4. else return \(\text{FALSE}\)

**SHARE-BOUNDARY** \((a, b)\)
1. \((I_a, I_b, M, N, O_a, O_b) \leftarrow \text{CLASSIFY} (a, b)\)
2. if \(I_a = 0\) and \(I_b = 0\) and \(M = 0\) and \(N \neq 0\)
   3. then return \(\text{TRUE}\)
   4. else return \(\text{FALSE}\)

**DISJOINT** \((a, b)\)
1. \((I_a, I_b, M, N, O_a, O_b) \leftarrow \text{CLASSIFY} (a, b)\)
2. if \(I_a = 0\) and \(I_b = 0\) and \(M = 0\) and \(N = 0\)
   3. then return \(\text{TRUE}\)
   4. else return \(\text{FALSE}\)
Chapter 6
Classification Algorithms

Common to the algorithms for each of the shape operations sum, product, difference and symmetric difference (see Algorithm 5.1) is the classification of the boundary segments of each of the operand shapes with respect to the other shape and the subsequent construction of the resulting shape from a set of boundary segments. The classification is a necessary component also in the algorithms that check whether one shape contains another, or whether two shapes overlap, share boundary or are disjoint (see Algorithm 5.2). In this chapter, we develop the classification algorithms for shapes of plane segments and shapes of volume segments. In Chapter 7, we develop the appropriate construction algorithms. When developing these algorithms, we pay particular attention to the resulting computational time and space complexities. A comparison of these results with similar algorithms for solid modeling is given in Chapter 8.

The classification of the boundary of a shape with respect to another shape requires splitting each boundary segment with respect to the other shape’s boundary, such that each split segment is wholly deemed inner, outer, same-shared or opposite-shared with respect to the other shape. The intersection of two non-parallel plane segments involves determining
those segments carried by the (infinite) line of intersection of both plane segment’s carriers that are inner or shared with respect to both plane segments. The following problem statement forms the basis of both tasks:

**Problem statement**  Classify an $n$-dimensional segment with respect to a co-equal shape in $U_{n+1}$ where the segment’s carrier lies within the shape’s carrier.

In the proofs to the algorithms below, we rely on the existence of an isomorphism between $U_{n,3}$ and a subset of $\wp(E^3)$ (see Section 4.2.2), however, implicitly. As such, we (may) operate on hyperplanes in $E^3$ and segments in $U_{n,3}$, simultaneously, as if they existed in the same space; e.g., the intersection of a line and line segment (see page 117) refers, specifically, to the point of intersection of this line with the point set, also in $E^3$, isomorphic to the specified line segment. We use $U_n$ to denote the algebra $U_{n,3}$.

### 6.1 Preliminaries

As spatial design is most often concerned with a three or lower dimensional space, we restrict ourselves the algebras in Table 6.1. The arithmetic of points in $U_0$ is trivial: Points combine only when they are identical; otherwise, they are disjoint. The arithmetic of line segments in $U_1$ is fully treated in Krishnamurti (1980) and Chase (1989). Krishnamurti (1992b, also 1992a) proves correctly the arithmetic of plane segments in $U_2$. Issues of time

<table>
<thead>
<tr>
<th>$U_{0,0}$</th>
<th>$U_{0,1}$</th>
<th>$U_{0,2}$</th>
<th>$U_{0,3}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_{1,1}$</td>
<td>$U_{1,2}$</td>
<td>$U_{1,3}$</td>
<td></td>
</tr>
<tr>
<td>$U_{2,2}$</td>
<td></td>
<td>$U_{2,3}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$U_{3,3}$</td>
</tr>
</tbody>
</table>

**Table 6.1** The algebras $U_{n,k}$ under consideration: The algorithms developed in this chapter are specific to the algebras $U_{2,3}$ and $U_{3,3}$. 

complexity are researched in Stouffs and Krishnamurti (1992). In Chapter 5, we considered a classification approach appropriate for algebras \( U_n (n > 0) \) and used it to prove correctly the arithmetic of maximal shapes in \( U_n \). Here, we develop the corresponding algorithms and determine their time and space complexities. Some results are repeated from Stouffs and Krishnamurti (1992).

In the sequel, we adopt an object-oriented algorithmic approach. That is, procedures such as \textsc{Sum}, \textsc{Product}, \textsc{Difference}, \textsc{Classify} and \textsc{Construct} are used independent of the algebra of their arguments, even though their implementation may differ with the algebra. For instance, the procedure \textsc{Sum} \((X, Y)\) returns the sum of two shapes \(X\) and \(Y\), where \(X, Y\) and the resulting shape belong to the same algebra. As a convention, we use capital letters to denote shapes, that is, sets of (maximal) segments, and lower case letters for single segments. \(^1\) We generally adopt the letters \(s\) and \(t\) (or \(S\) and \(T\)) to denote (shapes of) volume segments, and similarly, we use \(f\) and \(g\) for plane segments, \(k\) and \(l\) for line segments and \(p\) and \(q\) for points. We reserve the capital letters \(I, M, N, O\) to denote the classes of inner, same-shared, opposite-shared and outer segments, respectively, with a single subscript to denote the shape this class is a part of. For example, \(I_X\) denotes the class of inner segments, with respect to another shape, of the shape \(X\). The symbols +, − and \(\cup\) denote the set operations join, delete and merge, respectively, the latter operating on sorted sets. The operators + and − should not be confused with the shape operations of sum and difference which, in algorithmic usage, are represented by the procedures \textsc{Sum} and \textsc{Difference}.

The algorithms for the procedures \textsc{Sum}, \textsc{Product}, \textsc{Difference} and \textsc{Symmetric-Difference} (Algorithm 5.1) as well as \textsc{Contain}, \textsc{Overlap}, \textsc{Share-Boundary} and \textsc{Disjoint} (Algorithm 5.2), rely on procedures \textsc{Classify} and \textsc{Construct} in order to determine their result. The procedure \textsc{Classify} operates on two co-equal shapes in \(U_n\) \((n = 2, 3)\) and classifies the boundary segments of either shape with respect to the other shape. The classification algorithms for both algebras \(U_2\) and \(U_3\) are developed within this chapter. The procedure \textsc{Construct} takes as input a set of boundary (line or plane) segments, that defines a co-equal shape in \(U_n (n = 2, 3)\) and constructs this maximal shape.

\(^1\) Contrary to the algebras of shapes, we do not consider a segment to be a shape within the following algorithms. If \(x\) is a segment, then \(\{x\}\) denotes the corresponding shape. Correspondingly, we use \(\emptyset\) to denote the empty shape and \(0\) to denote the zero segment.
The construction algorithms for both algebras are developed in Chapter 7. Other procedures we consider merely serve to assist in the development of these algorithms. Below, we list some basic procedures that are common to the subsequent algorithms. The pseudo-code form and conventions used in the algorithms are due to Cormen et al. (1990). Comments within the algorithms are italicized.

- **INTERSECTION** \((\text{co-k}, \text{co-l})\) denotes the point \(p\) in \(U_0\), if any, isomorphic to the point of intersection of the lines in \(E^3\) defined by the co-descriptors \(\text{co-k}\) and \(\text{co-l}\).

- **INTERSECTION-2** \((l, \text{co-k})\) denotes the point \(p\) in \(U_0\), if any, isomorphic to the point of intersection of the carrier of \(l\) and the line in \(E^3\) defined by the co-descriptor \(\text{co-k}\), that satisfies the inequalities \(\text{tail}[l] \leq_c p \leq_c \text{head}[l]\).

- **LINE-SEGMENT** \((p, q)\) returns a line segment with endpoints \(p\) and \(q\).

- **INTERSECTION-LINE** \((\text{co-f}, \text{co-g})\) determines the co-descriptor of the line of intersection of the planes with co-descriptors \(\text{co-f}\) and \(\text{co-g}\).

- **NORMAL-PLANE** \((\text{co-l}, \text{co-f})\) determines the co-descriptor of a plane normal to the plane with co-descriptor \(\text{co-f}\) and through the line with co-descriptor \(\text{co-l}\).

- **PARALLEL** \((\text{co-x}, \text{co-y})\) compares two co-descriptors of segments or shapes in the same algebra and returns TRUE if they are either equal or represent parallel carriers and FALSE otherwise.

The procedures **SUM**, **PRODUCT** and **DIFFERENCE** as well as **CONTAIN**, **OVERLAP**, **SHARE-BOUNDARY** and **DISJOINT** on shapes of points in \(U_0\) are trivial. The algorithms for the same procedures applied to shapes of line segments in \(U_1\) are also well-known. We refer to Krishnamurti (1992a) for an overview of both. We remember that the procedures **SUM**, **PRODUCT** and **DIFFERENCE** take time linear in the size of their input, for shapes in \(U_0\) and \(U_1\). We consider the procedures **SUM**, **PRODUCT** and **DIFFERENCE** on shapes of plane segments in \(U_1\) dealing with non-equal shapes, also. The procedure **SORT** sorts a set of segments in correspondence to the total order defined on segments (see Section 4.1.1). Given a set of disjoint segments, the time complexity of **SORT** is dependent only on the size of the set and not on the sizes of the boundary shapes, i.e., equal to \(O(n \log n)\) where \(n\) is the number of segments. The procedure **MAXIMAL**, applied to an unsorted set of line segments that may share boundary but do not overlap, converts this set into the corresponding
maximal shape, also in time $O(n \log n)$ where $n$ is the number of segments. The following procedures all take time linear in their input size:

- **REMOVE-DUPLICATES** ($L$) removes pairs of duplicate or coinciding line segments from a shape in $U_1$.
- **REMOVE-MULTIPLES** ($L$) removes multiple occurrences of coinciding line segments from a shape in $U_1$.
- **REDUCE** ($X$) removes multiple occurrences of elements in a sorted set $X$.

All other procedures are defined upon use.

### 6.2 Classification of Line Segments

A first instance of the problem exists in classifying a line with respect to a shape in $U_2$.

---

**Lemma 6.1** Given a line $l$ and a co-equal shape $F$ in $U_2$, such that $l$ lies within the carrier of $F$, determining the inner and shared line segments with respect to $F$ that are carried by $l$ takes time $O(n \log n)$ and space $\Theta(n)$ where $n = |\text{boundary}[F]|$.

---

**Proof:** We consider a procedure **CLASSIFY-LINE**, the input to which consists of the co-descriptor $co-l$ of a line $l$ and a co-equal shape $F$ in $U_2$, where $l$ lies within the carrier of $F$. The results of the procedure are the classes of inner and shared line segments, with respect to $F$, with the given co-descriptor.

In Section 4.1.1 we defined an inside (and outside) to a boundary segment with respect to the shape it is a part of the boundary of. This is a natural consequence of the Jordan curve theorem (Henle, 1979), which states that any closed “path” partitions the (plane) space into two regions, one bounded, denoted the *inside* region, and one unbounded, the *outside* region.

Consider a co-equal shape $F$ in $U_2$ and an infinite line $l$ within the carrier of $F$. The boundary of $F$ defines an inside and outside region in the carrier plane, such that any part of $l$ that does not intersect the boundary of $F$ can be wholly deemed inner or outer with respect to $F$. Without considering degenerate cases, each point of intersection of $l$ and the boundary of $F$ is an endpoint to two disjoint parts (segments) of $l$, one of which is wholly deemed...
inner, while the other is wholly deemed outer, with respect to $F$. As such, the set of intersection points of $l$ and the boundary of $F$ defines an alternating sequence of inner and outer segments. Since the infinite ends of the line $l$ can be considered both outer and each intersection point alternates the classification, the total number of intersection points of $l$ and the boundary of $F$ must be exactly even.

Procedure 6.1 The points of intersection of the boundary of a co-equal shape $F$ and an infinite line $l$ within the carrier of $F$ defines an alternating sequence (if sorted) of inner and outer segments, starting with an inner segment.

Figure 6.1 illustrates the different cases for the intersection of a line $l$ and the boundary of a shape $F$ in $U_2$, where the line and shape are coplanar. Cases (b) through (e) are degenerate cases in which the point of intersection coincides with the endpoint of (at least) two boundary segments. Other degenerate cases may exist that are compositions of the cases. In order for these cases to be consistent with Procedure 6.1 for a set of intersection points on $l$, cases (a) and (b) must constitute a single point of intersection, cases (c) zero or two coincident points of intersection, cases (d) zero or two non-coincident points of intersection, and cases (e) again a single point of intersection.

Consider a line $l'$ parallel to the given line $l$ at an infinitesimal distance from $l$, within the carrier of $F$. For any boundary segment of the shape $F$ intersected by $l$ in one of its endpoints, the segment either intersects $l'$ in a single point that is not an endpoint or does not
intersect at all. Furthermore, whether there are zero or one intersection points depends solely on which side the boundary segment is with respect to the line \( l \), within the carrier of \( F \). Such a line \( l' \) always exists, and none of the degenerate cases for \( l \) can be a degenerate case for \( l' \). Figure 6.2 illustrates the resulting cases for \( l' \).

If we consider only those intersection points for \( l \) that correspond to the intersection points of \( l' \), and remove pairs of coincident points of intersection (case (c)), then all cases are consistent with Procedure 6.1. However, the class of inner segments determined from these points of intersection using Procedure 6.1, may contain shared segments. (The shared segments are the boundary segments of \( F \) with a co-descriptor equal to \( co-l \).) Then, the segments that are wholely deemed inner with respect to the shape \( F \) result from taking the difference of this class of inner segments with the class of shared segments previously determined.

Whether a single boundary segment has an intersection point with \( l \) does not depend on the particular case. As a result, we distinguish four basic cases (i through iv) for the intersection of a single boundary segment with a line, as illustrated in Figure 6.3. Any degenerate case is a composition of two or more of these four cases. When comparing the degenerate cases ii through iv, we notice that an intersection point is retained only if the boundary segment lies on one side of the line \( l \) and not on the other side, nor if it is a part of \( l \). It suffices to check only for one point on the segment, for instance, the other endpoint.

Figure 6.2 All degenerate cases are resolved by translating the line of intersection over an infinitesimal distance perpendicular to its axis.
Consider a line $l$ and a line segment $k$ with endpoints $p$ and $q$, such that $p$ lies on $l$ (see Figure 6.4). Let $\vec{d}_l$ denote the direction vector of $l$. The line $l$ and the carrier of $k$ define a plane $f$ in $E^3$ with equation $x \cdot \vec{n}_f + d_f = 0$. Consider the plane $g$ normal to $f$ and through $l$, where $f$ is the plane defined by $l$ and the carrier of $k$. Let $\vec{n}_g = \|\vec{d}_l \times \vec{n}\|$ and normal distance $d_g = -\vec{p} \cdot \vec{n}_g$. This plane divides $E^3$ into three regions of points $x$ for which the expression $x \cdot \vec{n}_g + d_g$ is either strict greater than zero, strict less than zero or equal to 0 (this last region is the plane $g$ itself). Let $co-g$ denote the co-descriptor representing $g$; let DOT-PRODUCT $(co[q], co-g)$ denote the result of the previous expression applied to point $q$, i.e., $q \cdot \vec{n}_g + d_g$. Then, the result of this procedure determines on which side of $g$, $q$ is located. Note that this expression is independent of $p$.

Given $co-g$, the co-descriptor of $g$, the following constitutes a simple condition to determine whether the point of intersection of a line segment $k$ is to be included:

$$\text{DOT-PRODUCT} (co[tail[k]], co-g) > 0 \text{ or } \text{DOT-PRODUCT} (co[head[k]], co-g) > 0.$$
6.2 Classification of Line Segments 121

This condition holds for all cases, degenerate or not: In case i, the endpoints of $k$ lie each on a different side, in case iv, the DOT-PRODUCT is zero for both endpoints. In cases ii and iii, one of the endpoints has a DOT-PRODUCT different from zero and one of the conditions must be positive while the other is negative.

**Complexity** (Algorithm 6.1): Let $n$ denote the number of maximal boundary segments of maximal segments of $F$, i.e., $n = |\text{boundary}[F]|$. Then, the sizes of both $M$ and $P$, and therefore $I$, are $O(n)$. $\text{SORT} (P)$ takes time $O(n \log n)$, the procedure $\text{DIFFERENCE}$ takes time linear in the input size, for shapes in $U_1$, and all other procedures take constant time. The $-$ operator is used to remove the first two elements of a set, which is achieved in constant time. Thus, the time complexity of the procedure $\text{CLASSIFY\text{-LINE}}$ with input size $n$ is $O(n \log n)$. 

---

**Algorithm 6.1**

```
CLASSIFY\text{-LINE} (co-l, F)
1   I ← M ← P ← ∅
2   co-g ← NORMAL\text{-PLANE} (co-l, co[F])
3   for each line segment $k \in \text{boundary}[F]$
4       do if $co[k] = co-l$
5          then $M ← M + \{k\}$
6        else $p ← \text{INTERSECTION\text{-2}} (k, co-l)$
7          if $p ≠ 0$
8             then if DOT-PRODUCT ($co[tail[k]], co-g$) > 0 or
9                DOT-PRODUCT ($co[head[k]], co-g$) > 0
10                then $P ← P + \{p\}$
11   \text{SORT} (P)
12   for each point $p \in P$
13       do if $p = \text{next}[p]$
14          then $P ← P – \{p, \text{next}[p]\}$ remove duplicate points
15   for each point $p \in P$
16       do $I ← I + \{\text{LINE\text{-SEGMENT} (p, next[p])}\}$
17       $P ← P – \{p, \text{next}[p]\}$
18   $I ← \text{DIFFERENCE} (I, M)$
19   return $(I, M)$
```
We can use the result of Lemma 6.1 when determining the intersection of two plane segments or shapes in $U_2$, each consisting of co-equal segments.

**Corollary 6.2** Given two shapes $F$ and $G$ in $U_2$, each consisting of co-equal segments, determining the line segments of intersection takes time $O(n \log n)$ and space $Θ(n)$ where $n = \lvert \text{boundary}[F] \rvert + \lvert \text{boundary}[G] \rvert$.

**Proof:** We consider a procedure INTERSECTION, the input to which consists of two shapes $F$ and $G$, each consisting of co-equal plane segments. The result is the shape of line segments of intersection from $F$ and $G$.

Consider the carrier planes of $F$ and $G$. If they are either identical or parallel, then the resulting shape of intersection is empty. Otherwise, we first define the (infinite) line of intersection and then use the procedure CLASSIFY-LINE to find the inner and shared segments of this line with either shape. The product of these two sets of segments results in the line segments of intersection of both shapes.

Consider two non-parallel planes $f$ and $g$ with equations $x \cdot n_f + d_f = 0$ and $x \cdot n_g + d_g = 0$, respectively. The line $l$ of intersection of $f$ and $g$ is uniquely defined by its direction vector $\vec{d}_l = \|n_f \times n_g\|$ and any point on $l$ (see Figure 6.5). Consider $\vec{d}_l$ ($l_x$, $l_y$, $l_z$), $n_f$ ($n_x^f$, $n_y^f$, $n_z^f$) and $n_g$ ($n_x^g$, $n_y^g$, $n_z^g$). If $l_x \neq 0$ then $l$ intersects the YZ-plane (with equation $x = 0$) at a point $p$ with coordinates $(0, y, z)$. This point necessarily lies on $f$ and $g$. Therefore, we have the set
Algorithm 6.2a

INTERSECTION \((F, G)\)

1. if PARALLEL \((\text{co}[F], \text{co}[G])\)

2. then return \(\emptyset\)

3. \(\text{co}-l \leftarrow \text{INTERSECTION-LINE} \((\text{co}[F], \text{co}[G])\) \) determine the line of intersection

4. \((I_F, M_F) \leftarrow \text{CLASSIFY-LINE} \(\text{co}-l, F\)\)

5. MAXIMAL \((M_F)\) \(I_F\) is necessarily maximal

6. \(I_F \leftarrow \text{SUM} \((I_F, M_F)\)\)

7. \((I_G, M_G) \leftarrow \text{CLASSIFY-LINE} \(\text{co}-l, G\)\)

8. MAXIMAL \((M_G)\) \(I_G\) is necessarily maximal

9. \(I_G \leftarrow \text{SUM} \((I_G, M_G)\)\)

10. return \(\text{PRODUCT} \((I_F, I_G)\)\)

of equations

\[
\begin{align*}
    n_y^f y + n_z^f z + d_f &= 0 \quad \text{with the solution} \quad y = \left( d_y^f n_z^g - d_y^g n_z^f \right) / \left( n_y^f n_z^g - n_y^g n_z^f \right) . \\
    n_y^g y + n_z^g z + d_g &= 0 \quad \text{with the solution} \quad z = \left( d_y^g n_z^f - d_y^f n_z^g \right) / \left( n_y^g n_z^f - n_y^f n_z^g \right) .
\end{align*}
\]

Let \(v = d_y n_f - d_f n_y\). The coordinates of \(p\) simplify to \(0, \frac{v_z}{l_z}, -\frac{v_y}{l_y}\). Similarly, if \(l_y \neq 0\) then \(l\) intersects the \(XZ\)-plane at a point \((-\frac{v_z}{l_z}, 0, \frac{v_y}{l_y})\), and if \(l_z \neq 0\) the intersection point with the \(XY\)-plane equals \(\left( \frac{v_y}{l_y}, \frac{v_x}{l_x}, 0 \right)\).

Complexity (Algorithm 6.2a): Let \(n\) denote the sum of the sizes of the boundaries of \(F\) and \(G\). The procedures PARALLEL and INTERSECTION-LINE both take constant time. The procedure CLASSIFY-LINE takes time \(O(n \log n)\) and returns a shape of size \(O(n)\). The procedures SUM and PRODUCT both take time linear in the size of their input, for shapes of line segments.

In a few cases we are rather interested in finding the intersection of a plane segment with a plane (e.g., see Algorithm 6.5). The procedure INTERSECTION-2 takes as input a plane segment and the co-descriptor representation of a plane (Algorithm 6.2a). The algorithm’s complexity is necessarily the same as for the procedure CLASSIFY-LINE.

Another instance of the above problem exists in classifying a boundary line segment with respect to a co-equal shape in \(U_2\).
Corollary 6.3  Given a boundary line segment $l$ and a co-equal shape $F$ in $U_2$, such that the carrier of $l$ lies within the carrier of $F$, classifying the segment $l$ with respect to $F$ (into the classes of inner, outer, same-shared and opposite-shared) takes time $O(n \log n)$ and space $\Theta(n)$ where $n = |\text{boundary}[F]|$.

Proof: We consider a procedure CLASSIFY-EDGE, the input to which consists of a line segment $l$ that is a boundary segment to some shape in $U_2$ and a co-equal shape $F$, also in $U_2$. The results of the procedure are the classes of inner, outer, same-shared and opposite-shared segments of $l$ with respect to $F$.

We can use the procedure CLASSIFY-LINE with arguments $\text{co}[l]$ and $F$ to determine the inner and shared line segments with co-descriptor $\text{co}[l]$, with respect to $F$. Taking the product of these inner segments with $l$ yields the class of inner segments of $l$ with respect to $F$. The class of same-shared segments results from the product of $l$ and the boundary segments $k$ of $F$ for which $\text{inside}[k]$ equals $\text{inside}[l]$. Similarly, the class of opposite-shared segments equals the product of $l$ and those boundary segments $k$ of $F$ for which $\text{inside}[k]$ differs from $\text{inside}[l]$. Finally, the class of outer segments of $l$ equals the difference of $l$ with the classes of inner and shared segments.

Complexity (Algorithm 6.3): Let $n$ denote the size of the boundary of $F$. The procedures PRODUCT, DIFFERENCE and MAXIMAL applied to shapes in $U_1$ all take time linear in the size of their arguments. Therefore, PRODUCT $\left(\{|k|\}, \{|l|\}\right)$ (lines 6-7) takes constant time while all other instances take time $O(n)$. It follows that the complexity of this algorithm is identical to the complexity for the procedure CLASSIFY-LINE.
6.2 Classification of Line Segments 125

Remark: The above algorithm can be improved by storing in $P$ only these intersection points that lie between $\text{tail}[l]$ and $\text{head}[l]$ while keeping count of all the intersection points left of or equal to $\text{tail}[l]$. The latter count defines whether the first segment starting at $\text{tail}[l]$ is deemed inner or outer with respect to $F$, while we use Procedure 6.1 to classify all subsequent segments. However, this improvement does not alter the complexity of the algorithm.

Algorithm 6.3

\begin{algorithm}
\textbf{CLASSIFY-EDGE} ($l, F$)
1 $I \leftarrow M \leftarrow N \leftarrow P \leftarrow \emptyset$
2 $co-g \leftarrow \text{NORMAL-PLANE} (\text{co}[l], \text{co}[F])$
3 \textbf{for} each line segment $k \in \text{boundary}[F]$
4 \hspace{1em} \textbf{do} if $\text{co}[k] = \text{co}[l]$
5 \hspace{2em} \textbf{then} if $\text{inside}[k] = \text{inside}[l]$
6 \hspace{3em} \text{then } M \leftarrow M + \text{PRODUCT} (\{k\}, \{l\})$
7 \hspace{3em} \text{else } N \leftarrow N + \text{PRODUCT} (\{k\}, \{l\})$
8 \hspace{2em} \text{else } p \leftarrow \text{INTERSECTION-2} (k, \text{co}[l])$
9 \hspace{1em} \textbf{if } p \neq 0$
10 \hspace{2em} \textbf{then} if $\text{DOT-PRODUCT} (\text{co}[\text{tail}[k]], co-g) > 0$ \textbf{or}\n11 \hspace{3em} \text{DOT-PRODUCT} (\text{co}[\text{head}[k]], co-g) > 0$
12 \hspace{2em} \textbf{then } P \leftarrow P + \{p\}$
13 \hspace{1em} \text{SORT} (P)
14 \hspace{1em} \textbf{for} each point $p \in P$
15 \hspace{2em} \textbf{do} if $p = \text{next}[p]$
16 \hspace{3em} \textbf{then } P \leftarrow P - \{p, \text{next}[p]\}$ \text{remove duplicate points}
17 \hspace{1em} \textbf{for} each point $p \in P$
18 \hspace{2em} \textbf{do } I \leftarrow I + \{\text{LINE-SEGMENT} (p, \text{next}[p])\}$
19 \hspace{1em} P \leftarrow P - \{p, \text{next}[p]\}$
20 \hspace{1em} \text{MAXIMAL} (M)$
21 \hspace{1em} \text{MAXIMAL} (N)$ \text{I is necessarily maximal}$
22 R \leftarrow \text{DIFFERENCE} (\{l\}, M)$
23 R \leftarrow \text{DIFFERENCE} (R, N)$
24 I \leftarrow \text{PRODUCT} (R, I)$
25 O \leftarrow \text{DIFFERENCE} (R, I)$
26 \textbf{return} (I, M, N, O)$
\end{algorithm}
A simple generalization of the above algorithm to classify the boundary of one shape with respect to another shape yields an $O(n^2 \log n)$ time complexity.

**Theorem 6.4** Given two co-equal shapes $F$ and $G$ in $U_2$, classifying the boundaries of $F$ and $G$ with respect to each other takes time $O((m + n) \log n)$ and space $\Theta(m + n)$ where $n = |\text{boundary}[F]| + |\text{boundary}[G]|$ and $m = O(n^2)$ is the total number of intersection points between boundary segments of $F$ and $G$.

**Proof:** We consider a procedure `CLASSIFY`, the input to which consists of two co-equal shapes $F$ and $G$ in $U_2$. The results of the procedure are the classes of inner and outer segments of each shape’s boundary with respect to the other shape and the classes of same-shared and opposite-shared boundary segments of both shapes. We use a plane-sweep algorithm (Bentley and Ottman, 1979) to determine the points of intersection of the boundary line segments of $F$ and $G$ and to classify the split segments with respect to the other shape.

Consider a vertical line sweeping the carrier plane of $F$ and $G$ from left to right. At any position, the sweep-line defines a cross section of the figure composed of the line segments of $F$ and $G$. Define the topology of a cross section to be the ordering of these segments intersecting the sweep-line, about this cross section. This topology remains invariant except at a finite number of transition points. These are the endpoints of the line segments as well as the points of intersection of segments from $F$ and $G$. Two consecutive transition points define a slice of the plane. The topology of each slice is encoded in the status of the sweep-line, at the time it sweeps this slice. This status is updated at each transition point.

Initially, consider only the endpoints as transition points. Given a line segment $l$ with endpoints $\text{tail}[l]$ and $\text{head}[l]$, we say that $l$ leaves from $\text{tail}[l]$ and arrives at $\text{head}[l]$. Then, two line segments intersect at a point $p$ only if the two segments are either consecutive, within the slice that immediately precedes transition point $p$, or separated by one or more segments arriving at $p$. In the former case, an inspection of the sweep-line status at the preceding transition point, after the update, reveals both line segments to be consecutive. In the latter case, $p$ constitutes an initial transition point. At each transition point, the status of the sweep-line is updated by removing the line segments arriving at $p$ and, subsequently,
6.2 Classification of Line Segments

inserting the line segments leaving from \( p \), in the correct order. The only segments affected by this update, with respect to possible forthcoming intersection points, are the two segments immediately above and below \( p \), not containing \( p \).

Nievergelt and Preparata (1982) distinguish four basic types of transition points, labelled \textit{start}, \textit{end}, \textit{bend} and \textit{intersection}. Figure 6.6 illustrates, for each of these four basic types, the pairs of line segments that need to be examined for forthcoming intersection points. These consist of the two segments immediately above and below \( p \), not containing \( p \), together with their immediate predecessor (below) and successor (above), respectively. As a result, at most two new transition points can be revealed at each transition point.

A plane-sweep algorithm operates on two basic structures: the task schedule, which contains the sorted list of transition points known so far, and the sweep-line status. Let \( H \) denote the structure representing the task schedule. Transition points are removed as they are processed, in order, and, upon determination, the intersection points are inserted. Thus, the structure \( H \) requires the abilities to extract the minimum element from and insert an element into, as well as check membership to \( H \). Such a structure may be considered a priority queue. Let \( D \) denote the structure representing the sweep-line status. At each transition point \( p \), the line segments arriving at \( p \) are removed from, and the line segments leaving from \( p \) are inserted into \( D \). Line segments intersecting at \( p \) are split and, subsequently, replaced by their right sub-segments, in reverse order. Then, the left sub-segments as well as any removed segments are classified into the classes of inner, outer, same-shared and opposite-shared segments with respect to either shape. Thus, the structure \( D \) is a dynamic set that requires the operations insert, delete, search and reverse to be supported.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure6.6.png}
\caption{The pairs of line segments to be examined for forthcoming intersection points, for each of the four basic types of transition points.}
\end{figure}
The line segments that define the status of the sweep-line, partition this line into inner and outer segments, according to Procedure 6.1. Similarly, the regions of the plane defined by these line segments, in the proximity of the sweep-line, can be classified into inside and outside regions, with respect to either shape, using Procedure 6.1. Then, a split line segment is deemed inner if it lies in a region classified as inside with respect to the shape that does not contain this segment, and outer if it lies in an outside region. Also, two coincident segments, belonging to different shapes, are deemed same-shared if either region, neighboring the coincident segments, is classified equal with respect to both shapes, and are deemed opposite-shared otherwise.

The procedure \textsc{Neighborhood-Segments} determines the segments immediately above and below the current transition point, not containing this point. It uses the procedure \textsc{Compare-Point-Wrt-Line} which compares the location of a point \( p \) with respect to a line (segment) \( l \). In particular, it returns the direction, i.e., sign, of the vector product of the direction vector \( \overrightarrow{d_l} \) of \( l \) and the vector \( \overrightarrow{qp} \) where \( q \) denotes \( \text{tail}(l) \), i.e., \( \overrightarrow{d_l} \times \overrightarrow{qp} \). This results in the value 0 if \( p \) lies on \( l \), a positive value if \( p \) lies “above” \( l \) and a negative value, otherwise (see Figure 6.7; see also Definition 7.1).

If we associate each region, as defined by two consecutive line segments in the status of the sweep-line, with the lower of these two bounding segments, then, we can augment the status of the sweep-line with the classifications of each region (whether it is inside or outside with respect to either shape). Let \text{Inside} and \text{Outside} denote the respective classifications and let \text{tag}(l, F) \] denote the classification of the region associated with \( l \), with respect to \( F \). The procedure \textsc{Classify-Segments} uses this information to classify each split segment with respect to the other shape.
Algorithm 6.4

CLASSIFY ($F, G$)
1 $R \leftarrow \emptyset$
2 $\text{shared} \leftarrow \text{FALSE}$
3 $H \leftarrow$ the endpoints of the boundary segments from $F$ and $G$,
   sorted lexicographically on their X- and Y-coordinate. For each point $p$ in $H$,
   let $L[p]$ be the set of boundary segments leaving from $p$.
4 $D \leftarrow [ -\infty, \infty ]$
5 while $H \neq \emptyset$
6 do $p \leftarrow \text{EXTRACT-MIN} (H)$
7 $(\text{low}, \text{high}) \leftarrow \text{NEIGHBORING-SEGMENTS} (D, p)$
8 for each $l \in D$ between low and high, low and high not included
9 do $m \leftarrow \text{succ}[l]$
10 if $\text{head}[l] = p$
11 then $k \leftarrow l$
12 DELETE ($D$, $l$)
13 else $k \leftarrow \text{CREATE-SEGMENT} (\text{tail}[l], p)$
14 $\text{tail}[l] \leftarrow p$
15 CLASSIFY-SEGMENT ($R$, $k$, $m$, $\text{shared}$)
16 REVERSE ($D$, $\text{succ}[\text{low}], \text{pred}[\text{high}]$)
17 for each $l \in L[p]$
18 do $\text{INSERT} (D, l)$
19 $q \leftarrow \text{INTERSECT} (\text{low}, \text{succ}[\text{low}])$
20 if not $\text{ELEMENT} (H, q)$
21 then $\text{INSERT} (H, q)$
22 $q \leftarrow \text{INTERSECT} (\text{high}, \text{pred}[\text{high}])$
23 if not $\text{ELEMENT} (H, q)$
24 then $\text{INSERT} (H, q)$
25 return $R$

NEIGHBORING-SEGMENTS ($D$, $p$)
1 $k \leftarrow \text{SEARCH} (D, p, \text{COMPARE-POINT-WRT-LINE})$
2 if $\text{COMPARE-POINT-WRT-LINE} (p, k) \leq 0$
3 then $k \leftarrow \text{pred}[k]$
4 $l \leftarrow \text{succ}[k]$
5 while $\text{COMPARE-POINT-WRT-LINE} (p, l) = 0$
6 do $l \leftarrow \text{succ}[l]$
7 return $(k, l)$
Algorithm 6.4 continued

```plaintext
CLASSIFY-SEGMENT (R, k, l, shared)
1 (IF, IG, M, N, OF, OG) ← R
2 if shared
3 then shared ← FALSE
4 else if tail[k] = tail[l]
5 then shared ← TRUE
6 if tag[k, F] = tag[k, G]
7 then Insert k into N.
8 else Insert k into M.
9 else if k ∈ F
10 then if tag[k, G] = INSIDE
11 then Insert k into IF.
12 else Insert k into OF.
13 else if tag[k, F] = INSIDE
14 then Insert k into IG.
15 else Insert k into OG.
16 R ← (IF, IG, M, N, OF, OG)
```

**Complexity** (Algorithm 6.4): The operations extract minimum element from (EXTRACT-MIN) and insert an element into (INSERT), as well as check membership to, on the structure $H$, take time bound $O(\log |H|)$, where $|H|$ denotes the size of $H$, when $H$ is implemented as a (heap or) balanced tree$^2$. The procedures INSERT, DELETE and SEARCH, on the structure $D$, take time bound $O(\log |D|)$, when $D$ is implemented as a balanced tree or splay tree (Sleator and Tarjan, 1985). By augmenting the tree structure to include predecessor ($\text{pred}[]$) and successor ($\text{succ}[]$) links, the procedure REVERSE takes time linear in the number of segments to be reversed.

Each boundary segment of $F$ and $G$ is inserted into and removed from $D$ exactly once. Let $n$ denote the total number of boundary segments of both $F$ and $G$. At each transition point, $D$ contains at most $n+2$ segments. Thus, insertion and deletion of the $n$ segments take time $O(n \log n)$. Let $m$ denote the number of intersection points between segments of $F$ and

---

2. We refer to Cormen et al. (1990) for a description of these and most other data structures we use. Otherwise, we note a specific reference.
segments of $G \ (m = O(n^2))$. Then, updating the sweep-line status at the $m$ intersection points takes time linear in $m$. Initializing the task schedule $H$ is achieved in time $O(n \log n)$. The total number of transition points is at most $m+n$ and, therefore, processing the task schedule takes time $O((m+n) \log (m+n))$. The procedure CLASSIFY-SEGMENT takes constant time. Thus, the entire plane-sweep, given $m$ and $n$ as defined above, takes time $O((m+n) \log n)$ with $m = O(n^2)$.

Note that the resulting shapes $I_F, I_G, O_F, O_G, M$ and $N$, corresponding to the classes of inner and outer (with respect to either shape), same-shared and opposite-shared segments, respectively, are not necessarily maximal. However, we know that no line segments, even from different classes, overlap (nor one contains the other) nor intersect (except at their endpoints). As such, the shapes (or sets of segments) $O_F \cup O_G \cup M, I_F \cup I_G \cup M, \ldots$ (see the procedures SUM, PRODUCT, DIFFERENCE and SYMMETRIC-DIFFERENCE in Algorithm 5.1), satisfy the conditions set forward in Theorem 7.1 (for the procedure CONSTRUCT when applied to line segments). Otherwise, it suffices to use the procedure MAXIMAL on each of the resulting shapes; this does not affect the asymptotic running time.

### 6.3 Classification of Plane Segments

Contrary to line segments, the boundary of a plane segment does not have a fixed size. However, given a shape $S$ in $U_3$, the total number of boundary line segments of boundary plane segments of $S$ is not arbitrary, but related to the number of boundary plane segments of $S$. The Euler-Poincaré equation specifies that

$$v - e + f = 2(s - g),$$

where $v$ denotes the number of vertices, $e$ the number of edges, $f$ the number of faces, $s$ the number of shells and $g$ the number of handles of a manifold solid. Given that each edge links exactly two vertices, and each vertex joins at least three edges, it follows that $v \leq 2/3 \ e$. Since $s \geq g$, we have that $v - e + f \geq 0$ or $f \geq e - v \geq e - 2/3 \ e \geq 1/3 \ e$. Also, since $v \geq 2(s - g)$, we have that $e + f \leq 0$ or $f \leq e$. Thus, $f = \Theta(e)$. We represent volume segments and shapes in $U_3$ as manifolds (see Section 4.2.3); therefore, we have that $n = \Theta(e)$, $m = \Theta(f)$ and, thus, $m = \Theta(n)$. 

Property  Given a shape $S$ in $U_3$, it holds that

$$\left| \text{boundary}[S] \right| = \Theta(\left| \text{boundary}[\text{boundary}[S]] \right|).$$

We consider (finite) plane segments only.

**Lemma 6.5**  Given a boundary plane segment $f$ and a shape $S$ in $U_3$, classifying the segment $f$ with respect to $S$ (into the classes of inner, outer, same-shared and opposite-shared) takes time $O(k n \log (k n))$ and space $O(k n)$ where $n = \left| \text{boundary}[S] \right|$ and $k = \left| \text{boundary}[f] \right|$.

**Proof:** We consider a procedure $\text{CLASSIFY-FACE}$, the input to which consists of a plane segment $f$ that is a boundary segment of some shape in $U_3$ and a (necessarily co-equal) shape $S$ in $U_3$. The results of the procedure are the classes of inner, outer, same-shared and opposite-shared segments of $f$ with respect to $S$.

A similar exposition can be made for the line segments of intersection of a plane with the boundary segments of a shape in $U_3$, as we made for the points of intersection of a line and a co-equal shape in $U_2$ (see proof of Lemma 6.1). Figure 6.8 illustrates the four possible cases of a plane segment intersecting a plane.

Consider a plane $f$ and a plane segment $g$ such that a boundary segment $l$ of $g$ has its carrier within $f$ (see Figure 6.9). Let $n_f$ and $n_g$ denote the normal vectors of $f$ and the carrier of $g$ and let $\vec{d}_l$ denote the direction vector of the carrier of $l$. Consider the unit vector $\nu$ perpendicular to both $\vec{d}_l$ and $n_g$ such that $\nu$ indicates the inside of $g$ with respect to $l$, i.e., $\nu = \text{inside}[l] \times n_g \vec{d}_l$. Then, the scalar product of $\nu$ and $n_f$ is a measure of the cosine of
angle between both vectors. That is, the sign of \( \mathbf{v} \cdot \mathbf{n}_f \) determines whether the inside of \( g \) with respect to \( f \) lies on one or the other side of \( f \). Also, the vector \( \mathbf{v} \) defines a plane \( h \) through \( l \) that is perpendicular to the carrier of \( g \). Let \( \text{co-} f \) and \( \text{co-} h \) denote the co-descriptors representing \( f \) and \( h \), respectively. Finally, let \( \text{DOT-PRODUCT} \ (\text{co-} h, \text{co-} f) \) denote the value of \( \mathbf{v} \cdot \mathbf{n}_f \).

**Complexity (Algorithm 6.5):** Let \( n \) denote the number of boundary line segments of boundary plane segments of \( S \). Let \( k \) denote the size of the boundary of \( f \). First, consider the construction of \( I \) from \( L \). For a given boundary segment \( g \) of \( S \), let \( n_g \) denote the size of the boundary of \( g \). Then, the procedure INTERSECTION-2 takes time \( O(n_g \log n_g) \), for each boundary segment \( g \), and the resulting sets \( L' \) and \( K \) have size \( O(n_g) \). Note that the line segments in \( K \) have received their inside information from \( g \) through the CLASSIFY-LINE procedure called from within INTERSECTION-2. Summed over all boundary segments \( g \) of \( S \), lines 7 through 13 take time \( O(n \log n) \) and result in a set \( L' \) of size \( O(n) \). Sorting the set \( L' \), removing duplicate line segments as well as constructing the set of plane segments \( I \), all take time \( O(n \log n) \). The resulting set \( I \) has size \( O(n) \), also.

Next, consider the sets of shared segments, \( M \) and \( N \). The product of \( f \) and \( g \) involves at most \( k n_g \) points of intersection and, therefore, takes time \( O(k n_g) \). The resulting sets \( M \) and \( N \) have size \( O(k n) \) and take time \( O(k n) \) to be assembled. The procedure MAXIMAL applied to sets of plane segments takes time \( O(k n \log (k n)) \) for input size \( O(k n) \) (see

**Figure 6.9** Given a plane segment \( g \) and a plane \( f \) through a boundary segment \( l \) of \( g \), we can determine the “side” of \( g \) with respect to \( f \) by computing the vector perpendicular to the direction vector of \( l \) and the normal vector of \( g \).
Algorithm 6.5

\begin{algorithm}
\textsc{Classify-Face} (f, S)
\begin{algorithmic}[1]
\State \textbf{L} $\leftarrow$ \textbf{M} $\leftarrow$ \textbf{N} $\leftarrow$ 0
\For {each plane segment \textit{g} $\in$ \textit{boundary}[S]}
\State \textbf{M} $\leftarrow$ \textbf{M} + \textbf{PRODUCT} ({\{\textit{g}\}}, {\{f\}})
\State \textbf{N} $\leftarrow$ \textbf{N} + \textbf{PRODUCT} ({\{\textit{g}\}}, {\{f\}})
\EndFor
\If {\textit{co}[\textit{g}] = \textit{co}[f]}
\State \textbf{L} $\leftarrow$ \textbf{L} + \textbf{PRODUCT} ({\{\textit{g}\}},{\{f\}})
\EndIf
\If {L' \neq 0}
\State \textbf{L} $\leftarrow$ \textbf{L} + L'
\EndIf
\For {each line segment \textit{l} $\in$ \textit{K}}
\State \textbf{co-h} $\leftarrow$ \textbf{NORMAL-PLANE} (\textbf{co}[\textit{l}], \textbf{co}[\textit{g}])
\If {\textit{inside}[\textit{l}] \cdot \textbf{DOT-PRODUCT} (\textbf{co-h}, \textbf{co}[f]) > 0}
\State \textbf{L} $\leftarrow$ \textbf{L} + \{\textit{l}\}
\EndIf
\EndFor
\State \textbf{sort} (\textbf{L})
\State \textbf{remove-duplicates} (\textbf{L})
\State \textbf{maximal} (\textbf{M})
\State \textbf{maximal} (\textbf{N})
\State \textbf{R} $\leftarrow$ \textbf{difference} (\{f\}, \textbf{M})
\State \textbf{R} $\leftarrow$ \textbf{difference} (\textbf{R}, \textbf{N})
\State \textbf{I} $\leftarrow$ \textbf{product} (\textbf{R}, \textbf{I})
\State \textbf{O} $\leftarrow$ \textbf{difference} (\textbf{R}, \textbf{I})
\State \textbf{return} (\textbf{I}, \textbf{M}, \textbf{N}, \textbf{O})
\end{algorithmic}
\end{algorithm}

Corollary 7.3). In lines 19 and 20 we compute the difference of \textit{f} with the classes of same-shared and opposite-shared plane segments. Since \textit{M} $\leq$ \textit{f}, \textit{N} $\leq$ \textit{f} and \textit{M} \cdot \textit{N} = 0, we can replace these lines with the following (partial) algorithm.

\begin{verbatim}
L $\leftarrow$ \textit{boundary}[M] $\cup$ \textit{boundary}[N]
\textbf{remove-duplicates} (\textbf{L})
L $\leftarrow$ L $\cup$ \textit{boundary}[f]
\textbf{remove-duplicates} (\textbf{L})
\textbf{R} $\leftarrow$ \textbf{construct} (\textbf{L})
\end{verbatim}

Since \textit{boundary}[f] has size \textit{k} and \textit{boundary}[M] and \textit{boundary}[N] both have size \textit{O}(\textit{k} \textit{n}), the preceding algorithm takes time \textit{O}(\textit{k} \textit{n} log (\textit{k} \textit{n})).
In lines 21 and 22 we compute the classes of inner and outer segments resulting from the product and difference, respectively, of $R$ and $I$. Using the characteristics of the classification approach, we can replace these lines with the following (partial) algorithm.

\[
\begin{align*}
(I_R, I_I, M, N, O_R, O_I) & \leftarrow \text{CLASSIFY} (R, I) \\
L & \leftarrow I_R \cup I_I \cup M \\
I & \leftarrow \text{CONSTRUCT} (L) \\
L & \leftarrow O_R \cup I_I \cup N \\
O & \leftarrow \text{CONSTRUCT} (L)
\end{align*}
\]

The complexity of the procedure \textsc{classify} applied to shapes in $U_2$ is dominated by the number of intersection points between boundary segments from both shapes. In this specific case, these intersections can only occur between boundary segments of $f$ and segments of $I$ or at the vertices of $S$. Thus, the number of intersection points is at most $O(kn)$ and the algorithm takes time $O(kn \log (kn))$.

\[\square\]

**Lemma 6.6** Given two co-equal shapes $S$ and $T$ in $U_3$, classifying the boundaries of $S$ and $T$ with respect to the other shape takes time $O(n \log n)$ and space $O(n)$ where $n = \lfloor \text{boundary}[S] \rfloor * \lfloor \text{boundary}[T] \rfloor$.

**Proof:** We consider a procedure \textsc{classify}, the input to which consists of two co-equal shapes $S$ and $T$ in $U_3$. The results of the procedure are the classes of inner and outer segments of each shape’s boundary with respect to the other shape and the classes of same-shared and opposite-shared boundary segments of both shapes.

**Complexity (Algorithm 6.6):** Let $n_S = \lfloor \text{boundary}[\text{boundary}[S]] \rfloor$ and $n_T = \lfloor \text{boundary}[\text{boundary}[T]] \rfloor$. For a given boundary segment $f$ of $S$, let $n_f$ denote the size of the boundary of $f$. Then, \textsc{classify-face} ($f, T$) takes time $O(n_f n_T \log (n_f n_T))$; summed over all boundary segments $f$ of $S$ this becomes $O(n_S n_T \log (n_S n_T))$, and similarly for the boundary plane segments of $T$ with respect to $S$. Since the sizes of all $I_S$, $I_T$, $M$, $N$, $O_S$ and $O_T$ are $O(n_S n_T)$, all sorting takes time $O(n_S n_T \log (n_S n_T))$, also.

\[\square\]

We can improve upon this result using the partitioning of the boundary segments into co-equal classes. Let $n$ denote the size of the boundary of a shape $S$ in $U_3$. Let $k$ denote the
136 Classification Algorithms

number of classes upon partitioning the boundary of $S$ into co-equal classes. Even though $k = O(n)$, it does not hold that $k = \Theta(n)$. Figure 6.10 illustrates a shape in $U_3$ where $k = \Omega(\sqrt[3]{n})$ (Karasick, 1988). The following theorem improves upon the result of Lemma 6.6 by using this distinction, as well as incorporating multiple applications of the INTERSECTION-2 procedure into a single plane-sweep algorithm.

### Theorem 6.7
Given two co-equal shapes $S$ and $T$ in $U_3$, classifying the boundaries of $S$ and $T$ with respect to the other shape takes time $O(k \log n)$ and space $\Theta(k n)$ where $n = |\text{boundary}[S]| + |\text{boundary}[T]|$ and $k = |\text{classes}[\text{boundary}[S]]| + |\text{classes}[\text{boundary}[T]]|$.

---

3. Karasick (1988) uses the same example for a similar purpose: his Star-Edge representation allows for a single face to consist of multiple polygons.
6.3 Classification of Plane Segments

Proof: We consider a procedure CLASSIFY, the input to which consists of two co-equal shapes $S$ and $T$ in $U_3$. The results of the procedure are the classes of inner and outer segments of each shape’s boundary with respect to the other shape and the classes of same-shared and opposite-shared boundary segments of both shapes.

The procedure PARTITION partitions the set of boundary segments into a set of co-equal classes. Each class can be parted further into two sub-classes, denoted $in[F]$ and $out[F]$, where $in[F]$ contains all boundary segments $s$ with $inside[s]$ equal to $+1$, and $out[F]$ contains all other segments (with $inside[s]$ equal to $-1$). Then, the class of same-shared boundary plane segments of $S$ and $T$ equals the sum of the products of $in[F]$ and $in[G]$, and $out[F]$ and $out[G]$, for all co-equal classes $F$ and $G$ in $S$ and $T$, respectively (line 19). Similarly, the class of opposite-shared boundary plane segments equals the sum of the products of $in[F]$ and $out[G]$, and $out[F]$ and $in[G]$ (line 20).

Each line of intersection between two classes of boundary segments, one from $S$ and one from $T$ (line 8), has to be classified with respect to either class to yield inner and shared segments (line 15). The procedure CLASSIFY-LINES has similar functionality as CLASSIFY-LINE (see proof of Lemma 6.1), except that it classifies multiple lines with respect to a single

Figure 6.10 A shape in $U_3$ consisting of $4^3/2 = 32$ segments (cubes), $4^3*3 = 192$ boundary plane segments, but only $3*(4+1) = 15$ co-equal classes of boundary segments.
co-equal shape. However its asymptotic running time can be improved upon by using a plane-sweep approach similar to the procedure CLASSIFY applied to sets of line segments (see proof of Theorem 6.4), when classifying all lines in a single sweep. Given a class $F$ from $S$ and the corresponding set of inner and shared segments for all intersecting classes from the shape $T$, this set, upon removing the shared segments on one side of the carrier of $F$ (see proof of Lemma 6.5), defines the boundary of the planar section of $T$ by this carrier. Then, the product of $F$ with the shape corresponding to this section determines the class of inner segments, including some shared segments. Let $I$ denote the shape corresponding to this section (lines 38 and 48). Consider the shape $R$ that is the difference of the class $F$ and the shared (with respect to classes of $T$) segments of $F$. The product of $R$ and $I$ determines the inner segments of $F$ with respect to $T$ (lines 41 and 51); the difference of $R$ and $I$ determines the outer segments of $F$ with respect to $T$ (lines 42 and 52). The same-shared and opposite-shared segments are gathered from the classes of $S$ or $T$ (lines 43 and 44). Thus, all classes of inner, outer, same-shared and opposite-shared segments can be found using simple arithmetic on shapes in $U_2$.

**Complexity** (Algorithm 6.7): Let $n$ denote the sum of the sizes of the boundaries of (the boundaries of) $S$ and $T$. Partitioning both sets of boundary segments into co-equal classes takes time $O(n \log n)$. Let $k$ denote the total number of classes. Determining the lines of intersection between classes of $S$ and $T$ takes time $\Theta(k^2)$; classifying these lines of intersection with respect to each of its defining classes takes time $O(k n \log n)$ in total (lines 6 through 15). The resulting sets of line segments total size $\Theta(k n)$. Determining the same-shared and opposite-shared line segments takes time linear in $n$ (lines 18-20). The procedure EXTRACT extracts the line segments with a given co-descriptor from a set of line segments that results from a call to the procedure CLASSIFY-LINES. Such can be done in $O(\log k)$ if this set is subdivided into sets of co-equal segments and these subsets are stored in a tree with depth $\log(k)$. Then, collecting the line segments of intersection for each class takes time $O(k^2 \log k + k n)$ in total (lines 16 through 34). Finally, constructing the classes of inner and outer segments for either shape as well as collecting the classes of same-shared and opposite-shared segments takes time $O(k n \log n)$. The number of intersection points $m$ is on the same order as the number of line segments of intersection. The sizes of the resulting
Algorithm 6.7

\textbf{CLASSIFY} \((S, T)\)

1. \(I_S \leftarrow I_T \leftarrow M \leftarrow N \leftarrow O_S \leftarrow O_T \leftarrow \emptyset\)
2. \(C_S \leftarrow \text{PARTITION} \ (\text{boundary}[S])\)
3. \(C_T \leftarrow \text{PARTITION} \ (\text{boundary}[T])\)
4. \textbf{for} each class \(F \in C_S + C_T\)
5. \hspace{1em} \textbf{do} \(\text{lines}[F] \leftarrow \text{same}[co[F]] \leftarrow \text{opp}[co[F]] \leftarrow \text{segments}[F] \leftarrow \emptyset\)

\textit{Determine line segments of intersection}

6. \textbf{for} each class \(F \in C_S\)
7. \hspace{1em} \textbf{do} \textbf{for} each class \(G \in C_T\)
8. \hspace{2em} \textbf{do} \(co-l \leftarrow \text{INTERSECTION-LINE} \ (co[F], co[G])\)
9. \hspace{3em} \textbf{if} \(co-l \neq 0\)
10. \hspace{4em} \textbf{then} \(\text{lines}[F] \leftarrow \text{lines}[F] + \{co-l\}\)
11. \hspace{4em} \(\text{lines}[G] \leftarrow \text{lines}[G] + \{co-l\}\)
12. \hspace{1em} \textbf{for} each class \(F \in C_S + C_T\)
13. \hspace{2em} \textbf{do} \text{SORT} \ (\text{lines}[F])
14. \hspace{1em} \text{REDUCE} \ (\text{lines}[F])
15. \hspace{1em} \textit{(inner}[F], \text{shared}[F]) \leftarrow \text{CLASSIFY-LINES} \ (\text{lines}[F], F)\)

\textit{Determine shared segments and collect line segments of intersection}

16. \textbf{for} each class \(F \in C_S\)
17. \hspace{1em} \textbf{do} \textbf{for} each class \(G \in C_T\)
18. \hspace{2em} \textbf{do} \textbf{if} \(co[F] = co[G]\)
19. \hspace{3em} \textbf{then} \(\text{same}[co[F]] \leftarrow \text{PRODUCT} \ (in[F], in[G])\)
20. \hspace{3em} \hspace{1em} + \text{PRODUCT} \ (out[F], out[G])
21. \hspace{3em} \(\text{opp}[co[F]] \leftarrow \text{PRODUCT} \ (in[F], out[G])\)
22. \hspace{3em} \hspace{1em} + \text{PRODUCT} \ (out[F], in[G])
23. \hspace{3em} \textbf{else} \(co-l \leftarrow \text{INTERSECTION-LINE} \ (co[F], co[G])\)
24. \hspace{3em} \hspace{1em} \textbf{if} \(co-l \neq 0\)
25. \hspace{4em} \textbf{then} \(\text{segments}[F] \leftarrow \text{segments}[F]\)
26. \hspace{4em} \hspace{1em} + \text{EXTRACT} \ (\text{inner}[G], co-l)\)
27. \hspace{3em} \(K \leftarrow \text{EXTRACT} \ (\text{shared}[G], co-l)\)
28. \hspace{3em} \textbf{for} each line segment \(l \in K\)
29. \hspace{4em} \textbf{do} \(co-h \leftarrow \text{NORMAL-PLANE} \ (co[l], co[G])\)
30. \hspace{5em} \textbf{if} \(\text{inside}[l] \cdot \text{DOT-PRODUCT} \ (co-h, co[F]) > 0\)
31. \hspace{6em} \textbf{then} \(\text{segments}[F] \leftarrow \text{segments}[F] + \{l\}\)
32. \hspace{5em} \(\text{segments}[G] \leftarrow \text{segments}[G]\)
33. \hspace{5em} + \text{EXTRACT} \ (\text{inner}[F], co-l)\)
34. \hspace{3em} \(K \leftarrow \text{EXTRACT} \ (\text{shared}[F], co-l)\)
Algorithm 6.7 continued

```
for each line segment \( l \in K \)
    do \( \text{co-h} \leftarrow \text{NORMAL-PLANE}(\text{co}[l], \text{co}[F]) \)
    if \( \text{inside}[l](\text{DOT-PRODUCT}(\text{co-h}, \text{co}[G])) > 0 \)
        then \( \text{segments}[G] \leftarrow \text{segments}[G] + \{l\} \)

Construct inner segments, collect shared segments and determine outer segments

for each class \( F \in C_S \)
    do \( \text{SORT}(\text{segments}[F]) \)
        \( I \leftarrow \text{CONSTRUCT}(\text{segments}[F]) \)
        \( R \leftarrow \text{DIFFERENCE}(F, \text{same}[\text{co}[F]]) \)
        \( I_S \leftarrow I_S + \text{PRODUCT}(R, I) \)
        \( O_S \leftarrow O_S + \text{DIFFERENCE}(R, I) \)
        \( M \leftarrow M + \text{same}[\text{co}[F]] \)
        \( N \leftarrow N + \text{opp}[\text{co}[F]] \)

for each class \( G \in C_T \)
    do \( \text{SORT}(\text{segments}[G]) \)
        \( I \leftarrow \text{CONSTRUCT}(\text{segments}[G]) \)
        \( R \leftarrow \text{DIFFERENCE}(F, \text{same}[\text{co}[G]]) \)
        \( I_T \leftarrow I_T + \text{PRODUCT}(R, I) \)
        \( O_T \leftarrow O_T + \text{DIFFERENCE}(R, I) \)

return \((I_S, I_T, M, N, O_S, O_T)\)
```

classes is on the order of the number of boundary segments \( n \) and the number of line segments of intersection \( m \), i.e., \( O(m + n) \) with \( m = O(k n) = O(n^2) \).

Note that, contrary to the procedure \textsc{Classify} when applied to line segments (see proof of Theorem 6.4), the resulting shapes \( I_F, I_G, O_F, O_G, M \) and \( N \), corresponding to the classes of inner and outer (with respect to either shape), same-shared and opposite-shared segments, respectively, are necessarily maximal.
Chapter 7
Construction Algorithms

Within the classification approach for operations on shapes in $U_n$ ($n = 2, 3$), the first step consists of splitting the boundary segments of either shape with respect to the other shape’s boundary, and of classifying the split segments into the classes of inner, outer, same-shared and opposite-shared segments for either shape with respect to the other.

Then, upon merging the appropriate classes together, the second step in the algorithm takes a given set of non-intersecting boundary segments and determines the resulting shape using a boundary traversal.

**Problem statement**  Given the boundary to a shape, construct the segments that make up that shape.

This construction also serves as a template for the conversion of the maximal representation to any other boundary representation. In the construction process, a graph is build containing topological information regarding adjacency that is often at the basis of a boundary representation.
Given an angle defined by three points (or two intersecting line or plane segments), we define clockwise and counterclockwise as follows:

**Definition 7.1** Given three points $p$, $q$, and $r$, the angle $\angle pqr$ is positive (including zero) or counterclockwise, if $\overrightarrow{pq} \times \overrightarrow{pr} \geq 0$, and negative or clockwise, otherwise.

Given a set of co-planar line segments, the vector product of the direction vectors of any two line segments is a vector with a fixed direction (except for the sign), that is, the notions of a clockwise and counterclockwise angle are independent of the two particular line segments under consideration.

### 7.1 Construction of Plane Segments

A first instance of the problem exists in extracting the simple boundaries from a set of non-intersecting boundary line segments.

**Theorem 7.1** Given a set $L$ of non-intersecting (except at their endpoints) line segments, that defines the boundary of a plane segment or shape $F$ in $U_2$, constructing the simple boundaries that define the plane segments of $F$ takes time $\Theta(n \log n)$ and space $\Theta(n)$ where $n = |L|$.

**Proof:** We consider a procedure $\text{EXTRACT-POLYGONS}$, the input to which consists of a set $L$ of line segments that forms the boundary of a shape in $U_2$. We assume the line segments in $L$ not to intersect (except at their endpoints) nor overlap, though they may coincide (i.e., be identical). The result of the procedure is a division of $L$ into subsets of line segments, each of which defines a simple boundary as a maximal shape in $U_1$.

In many cases there does not exist a unique solution to partitioning a set of line segments into (non-intersecting) simple boundaries. Figure 7.1 illustrates an example of a plane segment’s boundary that allows an interpretation as two outer boundaries or one outer and one inner boundary. Corresponding the definition of a segment (Definition 3.5), we adopt
7.1 Construction of Plane Segments

the interpretation of two outer boundaries. That is, we allow two polygons to share more than one endpoint only if they represent both outer or both inner boundaries.

Since we allow line segments to coincide, such cases as shown in Figure 7.2 are possible. The necessity to handle these cases follows from the procedure SPLIT applied to plane segments (see Algorithm 7.5). In the case of Figure 7.2(a), the result of the procedure EXTRACT-POLYGONS is a set of two simple boundaries that overlap. In the case of Figure 7.2(b), each pair of coinciding segments results in a trivial boundary cycle that is removed accordingly (see line 23 of procedure CYCLES in Algorithm 7.1).

The following outline of an algorithm formalizes this distinction. It subsumes that all simple boundaries are traversed in a counterclockwise manner:
Starting from the bottom left-most endpoint, proceed along the line segment that is closest to the bottom direction in a counterclockwise order about the endpoint.

At each endpoint on the path, proceed along the line segment that is closest to the last segment in a clockwise order about the endpoint.

Consider three line segments $a$, $b$ and $c$ with a common endpoint $p$ and no two line segments overlap. Let the ordering of the line segments about $p$ be represented as a triple, such that $(a, b, c)$ denotes a clockwise ordering and $(a, c, b)$ a counterclockwise ordering. All other permutations of $\{a, b, c\}$ are cyclic permutations of the above two. Therefore, we only consider the canonical cycles $(a, b, c)$ and $(a, c, b)$. Either cycle defines three angles about $p$ (the sum of which is $360^\circ$) only one of which can be greater or equal to $180^\circ$. Figure 7.3 illustrates clockwise and counterclockwise configurations in the cases when all angles are less than $180^\circ$ or a single angle is greater or equal to $180^\circ$. Table 7.1 formalizes these results. We conclude that three line segments $a$, $b$ and $c$ are configured clockwise about a common endpoint if at least two of the angles $\angle ab$, $\angle bc$ and $\angle ca$ are clockwise (cases i-iv), and are configured counterclockwise otherwise (cases v-viii).

Given an endpoint $p$, let $l_i$ ($i = 1, \ldots, n$) denote all line segments that have $p$ as an endpoint. For each segment $l_i$, let $q_i$ denote the other endpoint ($q_i \neq p$). Consider step 2 in the
algorithm: If the last segment is \( l_k \), then the continuation segment is \( l_j \) \((j \neq k)\) if and only if \((l_k, l_j, l_i)\) defines a clockwise ordering for all \(i \neq j, i \neq k\).

The bottom left-most endpoint is the greatest lower bound for the set of all endpoints under the order relation \( \leq_c \). Let \( p \) denote the bottom left-most endpoint. Let \( f \) denote the carrier plane of all line segments. Consider the line of intersection of \( f \) and a plane parallel to \( YZ \) through a point \( p \). If \( f \) is parallel to \( YZ \), then consider the line of intersection of \( f \) and a plane parallel to \( XZ \) through \( p \). This line of intersection defines the bottom direction \( b \) (see Figure 7.4). Let \( \vec{b} \) denote the direction vector of this line. Whatever its sign, one of the angles \( \angle bl_i \) and \( \angle l_i b \) is counterclockwise and the other is clockwise, for any line segment \( l_i \) with endpoint \( p \). Therefore, the starting segment is \( l_j \) if and only if \((b, l_j, l_i)\) defines a counterclockwise ordering, i.e., \( \angle l_i b \) is counterclockwise, for all \(i \neq j\).

The algorithm outlined above is a greedy algorithm that yields the smallest enclosed surface for the given starting segment. Figure 7.4 illustrates both steps in the algorithm. Let \( G \) denote the graph derived from the set of line segments by associating a vertex with each (unique) endpoint and an undirected edge with each line segment. Then, the simple boundaries correspond to simple cycles in the planar graph \( G \), which are extracted using a depth-first search on the graph. Starting from the bottom left-most vertex and proceeding as described above, a cycle or boundary is found when a vertex or point is reached that has been visited earlier in the traversal. If this vertex is the starting vertex, the traversal is concluded; other cycles may then be found by traversing the remaining graph, from a
(possibly) new vertex. Otherwise, the search is continued to find other cycles, until the starting vertex is reached. If more than one cycle is determined within a single traversal, all but the last one necessarily represent inner boundaries for the shape defined by the boundaries from this traversal (see Figure 7.5). However, since all are treated as outer boundaries, the insides of the inner boundaries’ line segments are inversed accordingly.\(^1\) Let \(V[G]\) denote the vertex set and \(E[G]\) the edge set. Each undirected edge is represented as

\[ l \]

\[ n_f \]

\[ b \]

\[ f \]

**Figure 7.4** Two examples illustrating the (a) start and (b) continuation steps in the outlined algorithm to extract the simple polygons from a set of boundary segments.

**Figure 7.5** An exemplar result of the procedure CYCLES, consisting of four simple cycles, of which all but one represent an inner boundary for the defined shape. Detailed is the traversal of a part of the cycles.

---

1. For each line segment \( l \) added to the cycle, the inside is indicated by the vector \( \vec{n}_f \times \vec{a}_l \) if \( l \) has retained the direction given in the counterclockwise traversal (lines 17-19).
two directed edge-halves, in opposite directions; each directed edge-half \((u, v)\) is defined as an entry in the adjacency list, denoted \(\text{Adj}[u]\), of the vertex \(u\).

**Complexity** (Algorithm 7.1): Let \(n\) denote the number of line segments. The procedure \text{STARTING-EDGE} takes time linear in the size of the adjacency list. Determining the continuation edge with respect to the current edge (procedure \text{CONTINUATION-EDGE}) can be achieved in constant time, so also the deletion of an edge from \(E[G]\) (i.e., from an adjacency list). All other steps in the procedure \text{CYCLES} take only constant time. Each time a vertex is pushed on the stack \(P\), an edge is traversed and both directed edge-halves are consequently removed from the graph. Each vertex popped from the stack results in a single line segment inserted into the cycle \(L\). Extracting a single cycle \(c\) takes time \(O(|c|)\), and summed over all cycles or simple boundaries, the total time taken is \(O(n)\).

Since each vertex links at least two edges, we have that \(|V[G]| \leq |E[G]|\). As such, the initialization of the graph (\text{ADJACENCY-GRAPH}) requires time \(\Theta(n \log n)\); the procedure \text{MAXIMAL} takes time \(O(n \log n)\) in total. \(\square\)

The next step is to distinguish the outer and inner boundaries from a set of simple boundaries and to construct the corresponding plane segments.

---

**Corollary 7.2** Given a set \(L\) of non-intersecting (except at their endpoints) line segments, that defines the boundary of a plane segment or shape \(F\) in \(U_2\),

\[\text{EXTRACT-POLYGONS}(L)\]

\begin{verbatim}
1 \(C \leftarrow \emptyset\)
2 \(G \leftarrow \text{ADJACENCY-GRAPH}(L)\) builds the graph corresponding to \(L\)
3 \textbf{for} each vertex \(v \in V[G]\) \textbf{in sorted order}
4 \hspace{1em} \textbf{do while} \(\text{Adj}[v] \neq \emptyset\)
5 \hspace{2em} \(C \leftarrow C + \text{CYCLES}(G, v)\)
6 \textbf{for} each shape \(L \in C\)
7 \hspace{1em} \text{MAXIMAL}(L)
8 \textbf{return} \(C\)
\end{verbatim}
Algorithm 7.1 continued

\[
\begin{align*}
\text{CYCLES } (G, u) \\
1 & \quad P \leftarrow C \leftarrow \emptyset \\
2 & \quad v \leftarrow u \\
3 & \quad \text{PUSH } (P, v) \\
4 & \quad (v, w) \leftarrow \text{STARTING-EDGE } (v) \\
5 & \quad \text{Adj}[v] \leftarrow \text{Adj}[v] - \{w\} \text{ remove directed edge } (v, w) \text{ from } E[G] \\
6 & \quad \text{while } P \neq \emptyset \\
7 & \quad \quad \text{do PUSH } (P, w) \\
8 & \quad \quad \quad (w, t) \leftarrow \text{CONTINUATION-EDGE } (w, (w, v)) \\
9 & \quad \quad \quad \text{Adj}[w] \leftarrow \text{Adj}[w] - \{v\} \\
10 & \quad \quad \quad \text{Adj}[w] \leftarrow \text{Adj}[w] - \{t\} \\
11 & \quad \quad \quad v \leftarrow w \\
12 & \quad \quad \quad w \leftarrow t \\
13 & \quad \quad \quad \text{if } w \in P \text{ a cycle is found} \\
14 & \quad \quad \quad \quad \text{then } L \leftarrow \emptyset \\
15 & \quad \quad \quad \quad \quad \text{while } P \neq \emptyset \text{ and } w \neq \text{top}[P] \\
16 & \quad \quad \quad \quad \quad \quad \text{do } l \leftarrow \text{LINE-SEGMENT } (\text{point}[l], \text{point}[\text{top}[P]]) \\
17 & \quad \quad \quad \quad \quad \quad \quad \text{if } \text{tail}[l] = \text{point}[l] \\
18 & \quad \quad \quad \quad \quad \quad \quad \quad \text{then } \text{inside}[l] = 1 \\
19 & \quad \quad \quad \quad \quad \quad \quad \quad \text{else } \text{inside}[l] = -1 \\
20 & \quad \quad \quad \quad \quad \quad \quad \quad L \leftarrow L + \{l\} \\
21 & \quad \quad \quad \quad \quad \quad \quad \quad t \leftarrow \text{POP } (P) \\
22 & \quad \quad \quad \quad \quad \quad L \leftarrow L + \{\text{LINE-SEGMENT } (\text{point}[l], \text{point}[w])\} \\
23 & \quad \quad \quad \quad \quad \quad \text{if } |L| > 2 \text{ if line segments do not coincide} \\
24 & \quad \quad \quad \quad \quad \quad \quad \text{then } C \leftarrow \{L\} + C \text{ insert } L \text{ at the front of the set} \\
25 & \quad \quad \quad \quad \quad \quad \text{Adj}[w] \leftarrow \text{Adj}[w] - \{v\} \\
26 & \quad \quad \quad \quad \text{for each shape } L \in \text{rest}[C] \\
27 & \quad \quad \quad \quad \quad \quad \text{do } \text{for each segment } l \in L \\
28 & \quad \quad \quad \quad \quad \quad \quad \quad \text{do } \text{inside}[l] \leftarrow -\text{inside}[l] \\
29 & \quad \quad \quad \quad \text{return } C
\end{align*}
\]

constructing the plane segments that make up \( F \) takes time \( \Theta(n \log n) \) and space \( \Theta(n) \)

where \( n = |L| \)

**Proof:** We consider a procedure \text{CONSTRUCT}, the input to which consists of a set \( L \) of line segments that forms the boundary of a shape in \( U_2 \). We assume the line segments in \( L \) not to
intersect (except at their endpoints) nor overlap, though they may coincide (see proof of Theorem 7.1). The result of the procedure is the shape in $U_2$ as a set of maximal plane segments.

**Complexity:** The procedure `CLASSIFY-POLYGONS` may be implemented using a plane-sweep to perform the classification in time $O(n \log n)$ where $n$ denotes the number of line segments in the subsets of $L$.

Alternatively, the algorithm could be modified in order to classify the cycles as they are returned by the procedure `CYCLES`. Consider the set of all simple boundaries extracted from the set $L$ of line segments and the shape defined by this set of boundaries. Then, because of the choice of the starting segment, the first cycle returned represents an outer boundary for the defined shape and all other cycles returned from the same traversal represent inner boundaries. Subsequently, consider the remaining set of simple boundaries and the shape defined by this set. Again, because of the choice of the starting segment, the first cycle returned represents an outer boundary for the defined shape and all other cycles returned from the same traversal represent inner boundaries. Furthermore, this outer boundary is either an outer boundary for the overall shape, in which case nothing changes, or an inner boundary, in which case the inner boundaries become outer boundaries for the overall shape. Thus, each set of cycles as returned by the procedure `CYCLES` can be classified at that moment, independently of any subsequently found cycles (see proof of Lemma 7.5 for a similar approach to the classification of simple boundaries in $U_2$). However, in the worst case, each traversal determines only a single cycle that has to be classified with respect to all of the previous found cycles. In such a case, the plane-sweep yields the best overall result.

---

**Algorithm 7.2**

```
CONSTRUCT (L)
1   L ← EXTRACT-POLYGONS (L)
2   return CLASSIFY-POLYGONS (L) using a plane-sweep
```
We can use the previous result when removing pairs of duplicate or coinciding segments from a set of plane segments as well as when determining the maximal shape given a set of plane segments that may share boundary (but do not overlap, nor one contains another).

**Corollary 7.3** Given a (sorted) set \( F \) of plane segments, removing pairs of coinciding segments and converting the resulting shape into its maximal representation takes time \( \Theta((m + n) \log n) \) and space \( \Theta(m + n) \) where \( n = |\text{boundary}[F]| \) and \( m \) denotes the number of intersection points between boundary segments of segments of \( F \).

**Proof:** We consider a procedure \textsc{Remove-Duplicates}, the input to which consists of a set \( F \) of (possibly overlapping) plane segments; the result is the shape in \( U_2 \) as a set of maximal plane segments, upon removing pairs of coinciding segments. We consider also a procedure \textsc{Maximal}, the input to which consists of a set \( F \) of plane segments that may share boundary, but do not overlap, nor one contains another; the result is the shape in \( U_2 \) in its maximal representation. We assume the set \( F \) sorted such that all co-equal segments are consecutive and all classes are in the correct order.

The procedure \textsc{Classify}, when applied to (sets of) plane segments, takes as arguments two co-equal (maximal) shapes in \( U_2 \) (see Algorithm 6.4). We know that the boundary segments of a maximal shape do not overlap (nor one contains another) nor intersect. As such, at any transition point (in the plane-sweep) at most two line segments intersect, one from each shape. However, no such assumption is included in the algorithm to the procedure \textsc{Classify}. For example, when examining the segments immediately above and below the transition point for forthcoming intersection points, no distinction is made as to whether both segments that may intersect belong to different shapes or not (lines 19 and 22 of \textsc{Classify}, page 130). Also, no restriction is made on the number of line segments that are split at each transition point (lines 8 through 15), nor the number of segments reversed (line 16). Therefore, if either or both shapes contain intersecting boundary segments, their points of intersection are found and inserted into the list of transition points and, subsequently, the respective segments split. The procedure \textsc{Split} is a variation on the procedure \textsc{Classify}; it takes as argument a single set of line segments and converts this into a set of non-
intersecting segments, at the same time extracting overlapping segments (as they are classified either same-shared or opposite-shared).

**Complexity** (Algorithm 7.3): Let \( n \) denote the number of line segments in \( L \), i.e., \( n = |L| \). Let \( m \) denote the number of intersection points between segments of \( L \) (\( m = O(n^2) \)). The procedure \( \text{SPLIT} \) has the same computational complexity as the procedure \( \text{CLASSIFY} \) when applied to shapes in \( U_2 \), i.e., \( O((m + n) \log n) \). The same time bound also holds for the procedure \( \text{CONSTRUCT} \).

It is possible to combine the functionality of the procedures \( \text{SPLIT} \) and \( \text{CONSTRUCT} \) into a single plane-sweep.

## 7.2 Construction of Volume Segments

Applied to the construction of volume segments the boundary traversal becomes a tree traversal process.

**Theorem 7.4** Given a set \( F \) of non-intersecting plane segments, that defines the boundary of a volume segment or shape \( S \) in \( U_3 \), constructing the simple boundaries that define the volume segments of \( S \) takes time \( O(Kn + n \log n) \) and space \( \Theta(n) \) where \( n = |F| \) and \( K \) is the number of simple boundaries.
Proof: We consider a procedure \textsc{extract-polyhedra}, the input to which consists of a set \( F \) of plane segments that forms the boundary of a shape in \( U_3 \). We assume the plane segments in \( F \) not to overlap nor intersect, except at their boundary line segments. Also, we assume the boundary line segments not to intersect nor overlap, but they may coincide, i.e., be identical.

The result of the procedure is a division of \( F \) into subsets of plane segments, each of which defines a simple boundary as a maximal shape.

Whereas the boundary traversal in \( U_2 \) is a linear process that determines cycles of boundary line segments, the corresponding traversal in \( U_3 \) is a tree traversal process. We define the horizon in a traversal to be the set of boundary line segments that have been reached but not yet completed. The following is an outline of the algorithm to create polyhedral shells of plane segments:

1. Starting from a left-most boundary line segment, proceed along the plane segment that is closest to the bottom direction.
2. Insert the boundary line segments of the current plane segment into the horizon, remove any duplicate line segments (in the horizon).
3. Take any line segment in the horizon, proceed along the plane segment that makes the smallest inside angle with the shell, about the line segment.
4. Proceed from 2.

Consider three plane segments \( f, g \) and \( h \) that share a boundary line segment \( l \), and no two of which overlap. For each plane segment, we determine a vector \( v \) perpendicular to the direction vector of \( l \) and the segment’s normal vector (see Figure 7.6). Then, the configuration (i.e., ordering) of the plane segments \( f, g \) and \( h \) is identical to the configuration of the vectors \( v_f, v_g \) and \( v_h \), about a point \( p \) on \( l \). That is, the angle \( \angle fg \) is equal to the angle defined by \( v_f \) and \( v_g \) about \( p \), which is counterclockwise if \( v_f \times v_g \geq 0 \).

Consider the set \( S \) of all boundary line segments that have the bottom left-most endpoint \( p \) as an endpoint. A left-most line segment \( l \) is a segment in \( S \) that makes the smallest angle with a plane \( a \) parallel to the \( YZ \) plane through \( p \). The angle of a line segment \( l \) with a plane \( a \) equals the angle between \( l \) and the normal projection of \( l \) on \( a \) (see Figure 7.7 (a)).

---

2. This, in order to ensure that every boundary line segment can be considered as a single entity that is either a part of the boundary of a segment or disjoint with that boundary.
7.2 Construction of Volume Segments 153

bottom direction is defined by a plane \( b \) through \( l \) and a line perpendicular to \( l \) within \( a \). Let \( n_b \) denote the normal vector of the plane \( b \). Let \( v_b \) denote the bottom direction as a vector

\[ v_f = \text{inside}[l, f] \quad \vec{n}_f \times \left[ \vec{d}_l \right] \]
\[ v_g = \text{inside}[l, g] \quad \vec{n}_g \times \left[ \vec{d}_l \right] \]
\[ v_h = \text{inside}[l, h] \quad \vec{n}_h \times \left[ \vec{d}_l \right] \]

Figure 7.6  The configuration of three plane segments \( f, g \) and \( h \) about a common boundary line segment \( l \) equals the configuration of the vectors \( v_f, v_g \) and \( v_h \) about a point \( p \) on \( l \), where \( v_f \) denotes a vector perpendicular to \( \vec{d}_l \) and \( n_f \), within \( f \) (idem for \( g \) and \( h \)). Counterclockwise (+) and clockwise (−) are defined relative to \( \vec{d}_l \).

Figure 7.7  Illustrations of (a) a left-most line segment \( l \) and (b) the bottom direction plane \( b \) with respect to \( l \). The plane \( a \) is a plane parallel to \( YZ \) that contains \( p \). \( i \) is a segment that makes the smallest angle with \( a \). The bottom direction vector \( v_b \) is perpendicular to the direction vector of \( l \) and the normal vector of \( a \).

3. Since the sine of the angle is proportional to the length of the vector product, it suffices to calculate the value \( \| \vec{d}_l \times \vec{i} \| \) for each line segment \( l \), in order to find the segment with the smallest angle.
for the purpose of calculating the angle with another plane segment, about $l$. Then, \[ v_b = \left\| \vec{n}_b \times \vec{d}_l \right\| = \left\| \vec{n}_u \times \vec{d}_l \right\|, \] where $\vec{d}_l$ and $\vec{i}$ denote the direction vectors of $l$ and the normal projection of $l$, respectively. The selection of the starting segment proceeds in a similar way as for the boundary traversal of plane segments (see proof of Theorem 7.1), that is, the selected segment makes a smallest angle, e.g., counterclockwise, with the bottom direction plane $b$. The inside of the shell under construction with respect to the starting segment is dependent upon the actual angle between the bottom direction plane and the starting segment $f$ and the direction of its normal vector $n_f$. That is, if $v_b \times v_f$ and $v_f \times n_f$ are both positive or both negative then $\text{inside}[f]$ equals 1, otherwise $\text{inside}[f]$ equals 1.

The determination of the continuation segment is dependent upon the inside of the plane segment it continues from. Consider the configuration of three plane segments $f$, $g$ and $h$, illustrated in Figure 7.6. Let $f$ represent the reference segment, i.e., the segment continued from. Let $c_f$ indicate the inside of the shell being constructed with respect to $f$, i.e., $c_f = \text{inside}[f] n_f$. If $c_f$ defines a counterclockwise angle from $v_f$, then the continuation segment is the segment closest to the reference in a counterclockwise order about $l$, e.g., segment $g$ in Figure 7.6. The inside of the shell with respect to $g$ is indicated by a vector $c_g$ clockwise from $v_g$; $\text{inside}[g]$ equals +1 if $n_g = c_g = v_g \times \left\| \vec{d}_l \right\|$ and equals 1 otherwise. Similarly, if $c_f$ defines a clockwise angle from $v_f$, then the continuation segment is the segment closest to the reference in a clockwise order about $l$ and the orientation of the inside of the shell with respect to the continuation segment is “counterclockwise”.

Consider the graph derived from the given set of plane segments by associating a vertex with each unique boundary line segment and an edge with each plane segment. Given that this set of plane segments constitutes the boundary of a shape in $U_3$, we say that the graph defines a boundary shape. We denote a *simple shell* any subgraph that defines a simple boundary; we denote a *composite shell* any subgraph that defines a boundary shape other than a simple boundary. Thus, the polyhedral boundaries that define the volume segments of the resulting maximal shape, correspond to simple shells in the graph, that are extracted using a tree-traversal process: Starting from a left-most vertex and proceeding as described in the algorithm outlined above, a shell is completed whenever the horizon is empty. However, similar to the boundary traversal in the construction of plane segments, a single traversal may not yield a simple shell, but a (composite) shell that defines a shape composed
of a single outer boundary and zero, one or more inner boundaries. Unlike the two-dimensional problem, the recognition of the simple shells that make up the current construction is not an obvious task (see Figure 7.9 and Figure 7.10).

Let the *multiplicity* of a vertex denote the number of edges it joins, that is, the number of plane segments, the boundary line segment is a part of the boundary of. The multiplicity of a vertex is, necessarily, even. Let a *partial shell* denote a maximal part of a shell that can be constructed by traversing vertices with multiplicity equal to 2, only (see Figure 7.8).

Therefore, the horizon of a partial shell defines a closed concatenation of boundary line segments corresponding to vertices with multiplicity greater than 2. Consider the partial shells resulting from a single application of the outlined algorithm on the given graph, we have that the combined horizon of these partial shells, under the set operation of symmetric difference, is empty. Let each partial shell be represented by a single (composite) edge and consider the (sub)graph of these edges joined by the common vertices in their horizons. Then, in a second traversal, within this (sub)graph, we consolidate partial shells that have a common vertex of multiplicity equal to 2 into new, partial or complete, simple shells (with an empty horizon). In the next cycle of the algorithm, we repeat both traversals on the

---

**Figure 7.8** Partial shells: (a) a composite shell and (b) an exploded view of the constituting partial shells. The emphasized partial shells are constructed in a single cycle.
remaining graph, and such until the entire graph is traversed and all simple shells are determined.

In order for this algorithm to succeed, it is imperative that, at each step in the second traversal, a vertex of multiplicity equal to 2 exists within the current (sub)graph. Suppose no such vertex exists. Then, at least some of the boundaries must form a meta-shell as illustrated in Figure 7.9. However, in case (a), because of the choice of the continuation edge, the construction of each outer boundary results in a separate cycle of the algorithm. Also, in case (c), because the outer boundary is defined by a single partial shell, the construction of this outer boundary results in a separate cycle of the algorithm, after which the construction is reduced to case (a).

Figure 7.9 Shells: (a) a shape defined by six outer boundary shells, (b) an exploded view of the same shape and (c) a shape defined by one outer and six inner boundary shells.
Let $G$ denote the graph derived from the set of plane segments by associating a vertex with each (unique) boundary line segment and an edge with each plane segment. Note that each edge links at least three vertices. Let $V[G]$ denote the vertex set and $E[G]$ the edge set. Each edge is represented as an entry in each of the adjacency lists of the vertices corresponding to the boundary line segments of the edge’s plane segment. Assume each adjacency list to be a cyclic list with the edge-halves ordered clockwise about the common vertex. A color scheme is used to distinguish the edges that partake in each stage of the construction. Initially, all edges are white. The first traversal constructs partial shells using only white edges. When an edge becomes part of the current construction its color is altered to gray. Upon completion of the first traversal, all gray edges compose the subgraph to be used during the consolidation process. As shells are completed, the composing edges are colored black. These edges no longer participate in the process.

The procedure SHELLS returns a set of boundary shells, of which a single outer boundary and zero, one or more inner boundaries; PARTIAL-SHELL extracts a partial shell as a composite edge containing two parts, a shell and a horizon. The shell is the shape of plane segments that is defined by the partial shell. The structure $H$ is a priority queue (prioritized on the vertices’ white or gray multiplicity) that represents the horizon under construction, and supports the operations INSERT, DELETE and MINIMUM. The procedure ORDERED-SET (line 19, PARTIAL-SHELL) converts the structure $H$ into an ordered set, representing the horizon. This supports the set operations $\cup$ (union), $\cap$ (intersection) and $\cup$ (symmetric difference). The procedure PRIORITY-QUEUE (line 19, SHELLS) rebuilds the set into a priority queue, using only the gray edges to determine the multiplicity for each vertex. For each vertex inserted in the horizon the last-edge field is updated to reference the edge to which the vertex belonged at the time of the insertion, as is necessary in order to determine the continuation edge at a later time.

$R$ denotes a registration table that supports the procedures REGISTER, UNREGISTER and RETRIEVE. Registration links an edge (whether composite or not) to a composite edge of which the former now makes a part. Registration removes the need for updating the edge for each of its vertices.
During the first traversal (lines 1-18), \( H \) constitutes the set of vertices that have been reached (an odd number of times) but not yet completed (an even number of times). Whenever a new partial shell is constructed, its horizon is added to \( H \) and duplicate vertices removed (through the symmetric difference operator). \( H \) constitutes the global horizon and guides the second traversal (lines 19-43). When two partial shells are consolidated, their horizons are merged and duplicate vertices removed. If the resulting horizon is empty, then a complete shell has been constructed. Otherwise, a new composite edge is created for the combined shell and horizon. As a result of the choice of the starting edge, we know that the first partial shell must define a part of an outer boundary (with respect to the shape defined by the current composite shell). The outer field of the composite edge encodes this information. As partial shells are consolidated, the outer information is passed on to the new composite edge. At any time, only one composite edge represents an outer shell.

**Complexity:** Consider a single execution of the procedure SHELLS. Let \( n \) denote the number of plane segments processed, with the number of boundary line segments equal to \( \Theta(n) \). The procedure CONTINUATION-EDGE takes constant time and so do the procedures COMPOSITE-EDGE, REGISTER, UNREGISTER, MINIMUM and the set operator +.

Consider the calls to the procedure PARTIAL-HELL: On a priority queue of size \( h \), the procedures INSERT and DELETE take time \( O(\log h) \). Since each vertex is inserted or deleted exactly once for each edge it joins, processing the priority queue \( H \) takes time \( O(n \log h) \).
Algorithm 7.4 continued

**SHELLS** \((G, u)\)

Extract partial shells (first traversal)

1. \(R \leftarrow S \leftarrow \emptyset\)
2. \(e \leftarrow \text{STARTING-EDGE}(u)\)
3. \(a \leftarrow \text{PARTIAL-SHELL}(G, e, R)\)
4. \(\text{outer}[a] \leftarrow \text{TRUE} \quad \text{first partial shell is a part of outer shell}\)
5. if \(\text{horizon}[a] = \emptyset\)
6. then for each plane segment \(f \in \text{shell}[a]\)
7. do UNREGISTER \((R, \text{edge}[f])\)
8. return \(\{\text{shell}[a]\} \quad \text{a single, simple shell}\)
9. \(H' \leftarrow H \leftarrow \text{horizon}[a]\)
10. for each vertex \(v \in H'\)
11. do \(e \leftarrow \text{CONTINUATION-EDGE}(v, \text{last-edge}[v])\)
12. \(a \leftarrow \text{PARTIAL-SHELL}(G, e, R)\)
13. if \(\text{horizon}[a] = \emptyset\)
14. then \(S \leftarrow S + \{\text{shell}[a]\}\)
15. for each plane segment \(f \in \text{shell}[a]\)
16. do UNREGISTER \((R, \text{edge}[f])\)
17. else \(H \leftarrow H \cup \text{horizon}[a]\)
18. \(H' \leftarrow H' \cup \text{horizon}[a]\)

Consolidate partial shells (second traversal)

19. PRIORITY-QUEUE \((H, \text{GRAY})\)
20. while \(H \neq \emptyset\)
21. do \(v \leftarrow \text{MINIMUM}(H)\)
22. \((d, e) \leftarrow \text{Adj}[v, \text{GRAY}]\)
23. \(d \leftarrow \text{RETRIEVE}(R, d)\)
24. \(e \leftarrow \text{RETRIEVE}(R, e)\)
25. \(H' \leftarrow \text{horizon}[d] \cap \text{horizon}[e]\)
26. for each vertex \(v \in H'\) remove common vertices from graph
27. do for each edge \(e' \in \text{Adj}[v, \text{GRAY}]\)
28. do if \(\text{RETRIEVE}(R, e') \in \{d, e\}\)
29. then \(\text{Adj}[v] \leftarrow \text{Adj}[v] - \{e'\}\)
30. if \(\text{Adj}[v, \text{GRAY}] = \emptyset\)
31. then \(\text{DELETE}(H, v)\)
32. \(F \leftarrow \text{shell}[d] + \text{shell}[e]\)
33. \(H' \leftarrow \text{horizon}[d] \cup \text{horizon}[e]\)
Algorithm 7.4 continued

if $H = \emptyset$

then if outer[$d$] $\lor$ outer[$e$]

then $S \leftarrow \{F\} + S$ place outer boundary in front of list

else $S \leftarrow S + \{F\}$

for each plane segment $f \in F$

   do UNREGISTER ($R$, edge[$f$])

else $a \leftarrow$ COMPOSITE-EDGE ($s$, $H'$)

   outer[$a$] $\leftarrow$ outer[$d$] $\lor$ outer[$e$]

   REGISTER ($R$, (d, a))

   REGISTER ($R$, (e, a))

for each shape $F \in \text{rest}[S]$

do for each segment $f \in F$

   do inside[$f$] $\leftarrow$ $-$inside[$f$]

return $S$

PARTIAL-SHELL ($G$, $e$, $R$)

1. $H \leftarrow \emptyset$ horizon as a priority queue
2. $a \leftarrow$ COMPOSITE-EDGE ($\{\text{segment}[e]\}$, $\emptyset$)
3. outer[$a$] $\leftarrow$ FALSE
4. REGISTER ($R$, (e, a))

for each line segment $l \in \text{boundary}[	ext{segment}[e]]$

do last-edge[vertex[$l$]] $\leftarrow$ e

   INSERT ($H$, vertex[$l$])

while $H \neq \emptyset$ and $|\text{adj}[\text{MINIMUM}(H)]| = 2$

do $v \leftarrow$ MINIMUM ($H$)

   $e \leftarrow$ CONTINUATION-EDGE ($v$, last-edge[$v$])

   Adj[$v$] $\leftarrow$ Adj[$v$] $\setminus$ {e, last-edge[$v$]}

   REGISTER ($R$, (e, a))

   shell[$a$] $\leftarrow$ shell[$a$] + {segment[e]}

   for each line segment $l \in \text{boundary}[	ext{segment}[e]]$

      do if vertex[$l$] $\in$ H

         then DELETE ($H$, vertex[$l$])

         else last-edge[vertex[$l$]] $\leftarrow$ e

            INSERT ($H$, vertex[$l$])

   horizon[a] $\leftarrow$ ORDERED-SET($H$)

return $a$
Converting the queue into an ordered set takes time $\Theta(h \log h)$ for an arbitrary ordering. Since $h = O(n)$, the combined time for all calls to \textsc{Partial-Shell} is $O(n \log n)$.

The procedure \textsc{Starting-Edge} is invoked only once and has $O(n)$ as an upper bound. Unregistration of the edges (plane segments) of the completed shell takes time $\Theta(n)$ in total. The set operations $\cup$, $\cap$ and $\cup$ take time linear in the size of the sets, $O(n)$. Let $K$ denote the number of incomplete, partial shells. As the set operations are executed possibly once for each partial shell, the combined time is $O(K \ n)$. Each time two partial shells are consolidated, retrieving any simple edge belonging to either partial shell takes one more step. Thus, in a dumb way, retrieving an edge from the registration table takes time $O(K)$. However, if the information is propagated backwards upon retrieving the final edge, the retrieval time can be reduced probably to almost constant time. Converting a set into a priority queue takes time $\Theta(n \log n)$. Thus, the total time taken to execute \textsc{Shells} once is $O(K \ n + n \log n))$. $K$ is a measure of the complexity of the resulting set of shells. In many instances, $K$ will equal 1; in the worst case, $K$ is in the order of the number of simple boundaries (see Figure 7.10).

Figure 7.10 A shape defined by one outer boundary and three inner boundaries. Each of these boundaries, with the exception of a single inner boundary, is constructed as two partial shells.
Consider the procedure \textsc{Extract-Polyhedra}: Let \( n \) denote the input size, let \( k \) denote a characteristic of the complexity of the traversed shells. The initialization of the graph requires time \( \Theta(n \log n) \). Extracting the shells takes time \( O(n \log n + K) \). The procedure \textsc{Maximal} takes time \( O(n \log n) \) in total.

The next step is to distinguish the outer and inner boundaries from a set of simple polyhedral boundaries and to construct the corresponding volume segments.

Lemma 7.5 Given a (sorted) set \( F \) of non-overlapping plane segments, that defines the boundary of a shape \( S \) in \( U_3 \), constructing the volume segments that make up \( S \) takes time \( O(Km + kn \log n) \) and space \( \Theta(m + n) \), where \( n = |F|, k = |\text{classes}[F]| \), \( m \) is the number of (non-boundary) line segments of intersection between segments of \( F \) and \( K \) is the number of simple boundaries of \( S \).

Proof: We consider a procedure \textsc{Construct}, the input to which consists of a set \( F \) of plane segments that forms the boundary of a shape in \( U_3 \). We assume that no two plane segments in \( F \) overlap, nor one contains the other. The result of the procedure is the shape in \( U_3 \) as a set of maximal volume segments.

After we have extracted the simple boundaries from the set \( F \), we need to distinguish the inner and outer boundaries and, subsequently, create the boundary shapes as they define the maximal volume segments of the resulting shape. We say that a boundary \( x \) encloses a boundary \( y \) if the shape defined by \( x \) contains the shape defined by \( y \), i.e., \( \Gamma(y) \subseteq \Gamma(x) \). Furthermore, we know that, if \( \Gamma(x) \cdot \Gamma(y) \neq 0 \) for two extracted boundaries \( x \) and \( y \), it follows that either \( \Gamma(x) \leq \Gamma(y) \) or \( \Gamma(y) \leq \Gamma(x) \).

Given the set of simple boundaries resulting from the procedure \textsc{Extract-Polyhedra}, consider the \textit{enclosure-tree} of these boundaries, defined as follows: Let \( \text{node}[x] \) denote the node in the tree representing boundary \( x \). If a boundary \( x \) encloses a boundary \( y \), then, \( \text{node}[x] \) is an ancestor of \( \text{node}[y] \) (and \( \text{node}[y] \) is a descendant of \( \text{node}[x] \)). If \( \text{node}[x] \) is the parent of \( \text{node}[y] \), then, boundary \( x \) encloses boundary \( y \) and any boundary that encloses \( y \) (with the exception of \( y \) itself) also encloses \( x \). Consider an imaginary root that encloses all boundaries. Then, all children of the root represent outer boundaries that are not enclosed...
by any other boundaries. Consequently, the grandchildren of the root represent inner boundaries that are enclosed by one of the previous boundaries. Let the level of a node denote the distance from the root and consider the root an inner boundary. Then, all nodes on even levels represent inner boundaries while all nodes on odd levels represent outer boundaries, with respect to the resulting shape. Thus, the set of boundaries from a single node on an odd level and its children, defines a maximal segment of this resulting shape. Figure 7.11 illustrates the enclosure-tree for a shape in $U_2$.

Consider two boundaries $x$ and $y$. Following Lemma 5.8, $\Gamma(x) \leq \Gamma(y)$ if and only if $I_{\Gamma(y)} = 0 \land O_{\Gamma(y)} = 0 \land N = 0$. However, in order to verify these conditions, we need to classify $x$ with respect to $\Gamma(y)$ and vice versa. The procedure CLASSIFY applied to $\Gamma(x) \leq \Gamma(y)$ takes time $O(n \log n)$ in the worst case, where $n = |x| + |y|$ (Lemma 6.6). Instead we use the following algorithm based on Procedure 6.1 in order to check enclosure for a single point within $x$, with respect to $y$.

1. Consider the carrier of any boundary line segment $l$ of a segment of $x$.
2. Determine the number of piercing points of this carrier with the segments of $y$ that are to the left to or equal to the tail of the line segment $l$.
3. Only if this number is odd, then, $y$ encloses $x$.

The piercing point of a line and a plane segment is the point of intersection, in $E^3$, of this line and the point set isomorphic to this plane segment. When determining the piercing point of a line with a plane segment, we distinguish whether the line pierces (the inside of) the plane segment, intersects (the inside of) the boundary of the plane segment, or contains an endpoint of a boundary segment of the plane segment; the latter two constitute “degenerate”
In order to resolve these degenerate cases, we define two reference directions with respect to the carrier line. That is, similar to the determination of the points of intersection of a line with the boundary of a shape in $U_2$ (see proof of Lemma 6.1), or, of a plane with the boundary of a shape in $U_3$ (see proof of Lemma 6.5), we consider translating the carrier line over an infinitesimal distance along directions perpendicular to this line.

Let $l$ denote the carrier line and $\vec{d}_l$ its direction vector. Let $r_1$ and $r_2$ denote the two reference direction vectors, with $\vec{d}_l$, $r_1$ and $r_2$ mutually perpendicular and $\|r_1\| = \|r_2\| = 1$. Consider the half-plane $f$ with boundary $l$ such that $r_1$ indicates the inside of $f$ with respect to $l$. The normal vector of $f$ equals the normalized vector product of $\vec{d}_l$ and $r_1$, i.e., $n_f = \|\vec{d}_l \times r_1\|$. We have that $n_f = \pm r_2$. Also, $\text{inside}[f]$ equals $+1$ if $n_f \times \vec{d}_l = r_1$ and $-1$, otherwise. This is illustrated in Figure 7.12.

Consider the case that $l$ intersects the inside of a boundary line segment $k$ of a plane segment $g$ of $y$, as illustrated in Figure 7.12. Consider the line $m$ of intersection of the carrier of $g$ and $f$. Assume $m$ is not co-linear with either $l$ or the carrier of $k$. Then, the piercing point of $l$
and \( g \) is a valid piercing point only if the (direction) vector \( m \) of the line \( m \) indicates the inside of \( g \) with respect to \( k \), that is, if \( \text{inside}[k] (n_g \times \vec{d}_k) \cdot m > 0 \). We have that \( m = \pm (n_f \times n_g), m \cdot r_1 > 0 \)

\[
v = \text{inside}[k] (n_g \times \vec{d}_k)
\]

Figure 7.13 An endpoint of a boundary segment \( k_j \) of a plane segment \( g \), where \( k_j \) makes the smallest angle with the line of intersection \( m \) of \( g \) and a reference half-plane \( f \), is a valid piercing point if (a) the direction vector of \( m \) indicates the inside of \( g \) with respect to \( k_j \), or (b) \( m \) coincides with the carrier of \( k_j \) and the second reference vector indicates the half-space with boundary \( f \) that contains \( g \).

Consider the case that \( l \) contains an endpoint \( p \) of a boundary line segment of a plane segment \( g \) of \( y \), as illustrated in Figure 7.13. Then, there exist at least two boundary line segments of \( g \) with endpoint \( p \). Let \( k_i, i \leq n \) denote all boundary line segments of \( g \) that have \( p \) as an endpoint. The scalar product of the direction vectors \( \vec{d}_k \) and \( m \) is a measure of the cosine of the angle \( \angle k_i \vec{d}_k m \) between the segments \( k_i \) and \( m \) (about \( p \), in the carrier plane of \( g \)).
If $\vec{d}_{kj} \cdot m$ is minimal for $i = j$ (1 ≤ $i$ ≤ $n$), then, the line segment $k_j$ makes the smallest angle with $m$, whether clockwise or counterclockwise. If the carrier of $k_j$ and $m$ do not coincide, i.e., $\vec{d}_{kj} \cdot m \neq 0$, then, $m$ indicates the inside of $g$ with respect to $k_j$. That is, the point $p$ is a valid piercing point if $\text{inside}[k_j] (n_g \times \vec{d}_{kj}) \cdot m > 0$. Let COMPARE-INSIDE-1 ($\text{col}[m], k_j, \text{col}[g]$) denote the result of the previous comparison (either TRUE or FALSE). If $(n_g \times \vec{d}_{kj}) \cdot m = 0$, then, $\vec{k}_j \cdot m = 0$ and $m$ coincides with the carrier of $k_j$. Thus, we use $r_2$ to determine the validity of the piercing point, i.e., $p$ is a valid piercing point if $\text{inside}[k_j] (n_g \times \vec{d}_{kj}) \cdot r_2 > 0$. Let COMPARE-INSIDE-2 ($\text{co-r}, k_j, \text{col}[g]$) denote the result of the previous comparison (either TRUE or FALSE), where the co-descriptor $\text{co-r}$ represents a plane with normal vector $r_2$.

We chose the following reference direction vectors: Given a plane segment $f$ with boundary segment $l$, the first reference vector indicates the inside of $f$ with respect to $l$ and, therefore, is perpendicular to the direction vector of $l$. The second reference vector is the normal vector of $f$: \[
\begin{align*}
\vec{r}_1 &= \text{inside}[l] (\vec{n}_f \times \vec{d}_l) \\
\vec{r}_2 &= \vec{n}_f
\end{align*}
\]

Before we can extract the polyhedral boundaries from the set of (plane) segments $F$, we have to ensure that no two plane segments intersect. The procedure SPLIT, when applied to a set of plane segments, determines the line segments of intersection for each plane segment with respect to all other segments and, subsequently, creates the sub-segments as defined by these line segments of intersection (see proof of Corollary 7.3 for the procedure SPLIT when applied to a set of line segments). In order to construct the simple boundaries of the sub-segments of a plane segment $f$, we use a single copy of the boundary segments of $f$ together with two copies of the line segments of intersection of $f$ as input to the procedure CONSTRUCT (see Algorithm 7.2).

After the plane segments are split and the polyhedral boundaries subsequently extracted, the enclosure-tree is constructed for the resulting simple boundaries. The sibling and child fields link a node with its siblings and children, respectively. The tree is initially empty. Each boundary is inserted in the tree in the order of appearance in the set of boundaries. We know from the algorithm for the procedure EXTRACT-POLYHEDRA (Algorithm 7.4) that every boundary is discovered before any boundaries it encloses.
The procedure \textsc{Enclosure} compares two simple boundaries \(F\) and \(G\) and returns \textsc{true} if \(G\) encloses \(F\) and \textsc{false}, otherwise. The procedure \textsc{Valid-Endpoint} checks the validity of a piercing point that coincides with an endpoint of a boundary segment. Similarly, the procedure \textsc{Valid-Intersection} checks the validity of a piercing point that coincides with an endpoint of a boundary segment. The procedure \textsc{Piercing-Point} determines the point of intersection of a line \(l\) and plane segment \(g\), in all but the above two cases. We use Procedure 6.1 to determine the validity of the piercing point, given the number of intersection points of an arbitrary line through the proposed piercing point and within the
Algorithm 7.5 continued

ENCLOSURE \((F, G)\)

\[
\text{odd} \leftarrow \text{FALSE} \quad \text{number of piercing points is either odd or even}
\]

\[
f \leftarrow \text{first}[F]
\]

\[
l \leftarrow \text{first}[\text{boundary}[f]]
\]

for each plane segment \(g \in G\)

\[
do \text{if } \text{co}[f] \neq \text{co}[g]
\]

\[
\text{then } k \leftarrow \text{first}[\text{boundary}[g]] \quad \text{check for intersection with boundary}[g]
\]

\[
p \leftarrow 0
\]

while \(k \neq 0\) and \(p = 0\)

\[
do \quad p \leftarrow \text{INTERSECTION-2} \ (k, \text{co}[l])
\]

\[
\text{if } p = 0
\]

\[
\text{then } k \leftarrow \text{next}[k]
\]

\[
\text{if } k \neq 0
\]

\[
\text{then if } p = \text{tail}[k] \text{ or } p = \text{head}[k] \quad \text{if intersection point is endpoint}
\]

\[
\text{then if } \text{VALID-ENDPOINT} \ (p, g, l, \text{co}[f])
\]

\[
\text{then odd } \leftarrow \text{not odd}
\]

\[
\text{else if } \text{VALID-INTERSECTION} \ (k, \text{co}[g], l, \text{co}[f])
\]

\[
\text{then odd } \leftarrow \text{not odd}
\]

\[
\text{else } p \leftarrow \text{PIERCING-POINT} \ (\text{co}[l], g)
\]

\[
\text{if } p = 0
\]

\[
\text{then odd } \leftarrow \text{not odd}
\]

\[
\text{return odd}
\]

EXTRACT-SEGMENTS \((t)\)

\[
S \leftarrow \emptyset
\]

\[
\text{while } t \neq \text{NIL}
\]

\[
do \ u \leftarrow \text{child}[t]
\]

\[
R \leftarrow \{\text{shape}[r]\}
\]

\[
\text{while } u \neq \text{NIL}
\]

\[
do \ S \leftarrow S + \text{EXTRACT-SEGMENTS} \ (\text{child}[u])
\]

\[
\text{for each segment } x \in \text{shape}[u]
\]

\[
\text{do } \text{inside}[x] \leftarrow \text{inside}[x]
\]

\[
R \leftarrow R + \{\text{shape}[u]\}
\]

\[
u \leftarrow \text{sibling}[u]
\]

\[
S \leftarrow S + \text{SEGMENT} \ (R)
\]

\[
t \leftarrow \text{sibling}[t]
\]

\[
\text{SORT} \ (R)
\]

\[
\text{return } S
\]
Algorithm 7.5 continued

**VALID-EPNDPT** (*p*, *g*, *l*, *co-f*)
1. \(co-m \leftarrow \text{INTERSECTION-LINE}(co[g], co-f)\)
2. \(j \leftarrow 0\)
3. for each line segment \(k \in \text{rest}[\text{boundary}[g]]\)
4. do if \(\text{tail}[k] = p\) or \(\text{head}[k] = p\)
5. then if \(j = 0\)
6. then \(j \leftarrow k\)
7. else if \(\text{DOT-PRODUCT}(co[k], co-m) < \text{DOT-PRODUCT}(co[j], co-m)\)
8. then \(j \leftarrow k\)
9. if \(\text{DOT-PRODUCT}(co[j], co-m) = 0\)
10. then return \(\text{COMPARE-INSIDE-2}(co-f, j, co[g])\)
11. else return \(\text{COMPARE-INSIDE-1}(co-m, j, co[g])\)

**VALID-INTERSECTION** (*k*, *co-g*, *l*, *co-f*)
1. \(co-m \leftarrow \text{INTERSECTION}(co-g, co-f)\)
2. if \(co[k] = co-m\)
3. then return \(\text{COMPARE-INSIDE-2}(co-f, k, co-g)\)
4. else return not (\(\text{COMPARE-INSIDE-1}(co-m, k, co-g)\)) \(\text{xor} \text{COMPARE-INSIDE-1}(co-m, l, co-f)\)

**PIERCING-POINT** (*co-l*, *f*)
1. \(odd \leftarrow \text{FALSE}\)
2. \(m \leftarrow \text{PIERCING}(co-l, co[f])\)
3. \(co-g \leftarrow \text{NORMAL-PLANE}(co-l, co[f])\)
4. \(co-m \leftarrow \text{INTERSECTION}(co[f], co-g)\)
5. for each line segment \(k \in \text{boundary}[f]\)
6. do if \(co[k] \neq co-l\)
7. then \(q \leftarrow \text{INTERSECTION-2}(k, co-m)\)
8. if \(q \neq 0\) and \(q \leq c_p\)
9. then \(odd \leftarrow \text{not} odd\)
10. if odd
11. then return \(p\)
12. else return \(0\)

carrier of \(g\) (e.g., the line of intersection of this carrier and a normal plane through the line \(l\)), with the boundary segments of \(g\). Finally, the tree is traversed and the boundaries extracted.
and grouped as they define the maximal segments of the resulting shape (EXTRACT-SEGMENTS).

**Complexity** (Algorithm 7.5): Let \( n \) denote the size of (the boundary of) \( F \). Consider the procedure \textsc{Split}: Determining the line segments of intersection between the classes of plane segments of \( F \) and \( G \) takes time \( \mathcal{O}(k n \log n) \) where \( k \) is the total number of classes of \( F \) and \( G \) (lines 4-8). Constructing the plane segments from the set of boundary segments and line segments of intersection (two copies) of a class of segments takes time \( \mathcal{O}((m + n) \log n) = \mathcal{O}(k n \log n) \), where \( m \) is the number of line segments of intersection, i.e., \( m = \mathcal{O}(k n) \) (lines 9 through 15). This bound is also the overall complexity for the procedure \textsc{Split}. The resulting size of \( F \) is \( \mathcal{O}(m + n) \).

Extracting the simple boundaries (EXTRACT-POLYHEDRA) from \( F \) takes time \( \mathcal{O}((m + n) (\log n + K)) \), where \( K \) denotes the number of resulting simple boundaries, and results in a set \( R \) of size \( \mathcal{O}(m + n) \). Consider the procedure ENCLOSURE: Determining the point(s) of intersection of a line segment with the boundary of a plane segment takes time linear in the size of this boundary (lines 6-11). This time complexity holds also for the procedure PIERCING-POINT, as well as VALID-ENDPOINT and VALID-INTERSECTION (constant time, actually). Since, we repeat this computation for each segment of \( G \), the procedure ENCLOSURE takes time \( \mathcal{O}(\alpha) \), where \( \alpha \) denotes the sum of the sizes of both simple boundaries. Upon insertion of a boundary into the tree \( T \) (line 5, \textsc{Construct}), each boundary already in the tree may need to be examined for possible enclosure, in the worst case. Therefore, building the tree takes time \( \mathcal{O}(K (m + n)) \) for \( K \) simple boundaries. Extracting the segments (EXTRACT-SEGMENTS) from \( T \) takes time \( \mathcal{O}(K \log K + m + n) \). The resulting time complexity of the procedure \textsc{Construct} is \( \mathcal{O}(k n \log n + K m). \) ❑

This algorithm may be improved by considering the simple boundaries that result from EXTRACT-POLYHEDRA, grouped as they are returned from separate calls to the procedure SHELLS (Algorithm 7.4). This procedure returns a set of one or more simple boundaries of which all but the first are, relatively, inner boundaries with respect to the first one. As such, these “inner” boundaries no longer need to be classified. However, in the worst case, each call to the procedure SHELL returns a single boundary and the time complexity of the procedure \textsc{Construct} remains the same.
We can use the previous result when determining the maximal shape corresponding to a set of volume segments that may share boundary, but do not overlap nor one contains another.

**Algorithm 7.6**

```
MAXIMAL (S)
  1  SPLIT (boundary[S])
  2  REDUCE (boundary[S])
  3  R ← EXTRACT-POLYHEDRA (boundary[S])
  4  T ← NIL
  5  for each shape F ∈ R in order
  6     do INSERT (T, F, ENCLOSURE)
  7  S ← EXTRACT-SEGMENTS (T)
```

The time complexity of the procedure MAXIMAL is necessarily the same as for CONSTRUCT, i.e., \( O(K m + k n \log n) = O(k n (\log n + K)) \), where \( n = |\text{boundary}[S]| \), \( k = |\text{classes}[\text{boundary}[S]]| \), \( m \) is the number of line segments of intersection between segments of \( \text{boundary}[S] \) and \( K \) is the number of simple boundaries of \( S \).
The representation of figures (shapes, solids, point sets or otherwise) and efficient algorithms for their manipulation have long interested researchers in computer-aided design and modeling. The literature pertaining to Boolean set operations on polygons and polyhedra, in particular, falls into two main research camps, respectively, *computational geometry* and *solid (or geometric) modeling*, each addressing its own interests with respect to this topic. The former expounds efficiency, but limits itself to dealing with convex polygons or polyhedra. The latter unanimously adopts a set-membership classification approach but falls short on efficient algorithms.

Preparata and Shamos (1985) summarize the views within computational geometry. A typical problem is the intersection of two convex polygons: Shamos and Hoey (1976) subdivide convex polygons into quadrilaterals (or trapezoids) and employ a plane-sweep to find the intersection (or union or difference) of both polygons. The resulting algorithm runs in time linear in the number of vertices (or edges) of the two polygons. O’Rourke et al. (1982) determine the intersection of two convex polygons by concurrently traversing both polygons, also in linear time. Hertel et al. (1984) conceive an approach for convex
polyhedra that combines a space-sweep to find all intersection points of both polyhedra followed by a single boundary traversal, and that runs in $O(n \log n)$ time, where $n$ equals the total number of vertices of both polyhedra. Preparata and Shamos give a lower bound on the time required to find the intersection of two star-shaped polygons, which is on the order of the number of intersection points of both polygons and thus $\Omega(n^2)$ in the worst case.

Nievergelt and Preparata (1982) describe two plane-sweep algorithms to process the regions of the plane defined by intersecting figures. The first algorithm finds the regions defined by a self-intersecting polygon and runs in time $O((m+n) \log n)$ where $n$ denotes the number of vertices or edges of the polygon and $m$ denotes the number of intersection points. The second algorithm computes all $m$ intersections of two convex maps (embedded planar graphs with convex regions) with a total of $n$ points in time $O(m + n \log n)$. Mairson and Stolfi (1988) show that, given two sets $A$ and $B$ each consisting of non-intersecting line segments, such that $|A| + |B| = n$, time $O(m + n \log n)$ suffices to report all $m$ intersections of segments of $A$ with segments of $B$. This yields an algorithm to compute the intersection of two polygons with the same time complexity.

Some of these results can and have been used in the development of the algorithms in Chapter 6 and Chapter 7. Below, we recap the asymptotic time complexity results we achieved and compare these with similar results in the literature. We conclude with a discussion of the robustness of the computations.

### 8.1 Computational Complexity

In Chapter 5 through Chapter 7 we have developed algorithms for the arithmetic operations of sum, product, difference and symmetric difference on shapes using a classification approach. That is, given two operand shapes, the boundary segments of either shape are classified with respect to the other shape and, subsequently, the resulting shape is constructed using a boundary traversal on a set of boundary segments resulting from the merging of the appropriate classes. Here we adopt a different approach to (the complexity of) the arithmetic operations that allows us to give a general time bound on the corresponding algorithms given a bound on identical operations when applied to pairs of co-equal segments. Subsequently, we summarize the asymptotic complexities found for the
8.1 Computational Complexity

arithmetic operations and relations on shapes and compare these with similar results pertaining to regularized set operations on polygons or polyhedra given in the literature.

8.1.1 Time Complexities

Let combine, common and complement denote the respective operations of sum, product and difference applied to a pair of co-equal segments. Assume that the running time of each of these operations, as well as the shape relations contain, overlap, share boundary and disjoint, when applied to a pair of co-equal segments, is bounded by some function \( f(n) \), where \( n \) is a measure of the size of the boundaries of the segments. The particular function \( f \) is, obviously, dependent on the algebra under consideration. Then, the running times of the shape operations are asymptotically bound by some function of \( f(n) \).

Theorem 8.1 The shape operations of sum, product, difference and symmetric-difference applied to two co-equal shapes \( a \) and \( b \) have asymptotic time complexities at most quadratic in the number of maximal segments \( N \) of both \( a \) and \( b \), i.e., \( O(N^2 f(n)) \), where \( n \) is a characteristic size of the boundaries of the maximal segments of \( a \) and \( b \) and the function \( f \) denotes the asymptotic time complexity of the shape operations and relations when applied to two co-equal segments.

Proof: Given two co-equal shapes \( a \) and \( b \) and their representations as two sets of co-equal segments, determining the sum, product or difference of both shapes, may require an operational comparison of each segment from \( a \) with each segment from \( b \), in the form of a single operation, namely, combine, common or complement, respectively, and a test for up to four relations, namely, contain, overlap, share boundary and disjoint. Therefore, each comparison requires at most five operations and relations that each, as well as combined,

1. We write \( f \) to be a function of a single variable, though in practice its value may depend on more than one variable, e.g., the size of the boundaries \( n \) and the number of classes of boundary segments \( k \).
2. Actually, four since the relations disjoint, overlap, share boundary and disjoint are mutually exclusive relations and exactly one relation applies to any pair of segments or shapes (see Section 2.2 as well as Section 3.2.2)
take time $O(f(n))$. Then, comparing the representational sets of $a$ and $b$ takes time $O(N^2 f(n))$.

The same time bound holds for two shapes that are not necessarily co-equal: Consider two shapes $a$ and $b$ and, for either shape, the classes of co-equal segments. Let $N_k$ denote the total number of segments in each pair $k (k = 1, \ldots)$ of co-equal classes (one from $a$ and one from $b$). Let $N$ denote the total number of segments, i.e., $N = \sum N_k$. We know that the comparison of two co-equal classes takes time $O(N_k^2 f(n))$, for each $k$. Since segments can only combine (under the operation of sum, product, difference or symmetric difference) if they are not disjoint (e.g., see Lemma 3.6) and, therefore, belong to the same class, the total time for the comparison of the representational sets of segments for the shapes $a$ and $b$ is bound by $O(\sum N_k^2 f(n)) = O(N^2 f(n))$. This is also the asymptotic upper bound on the running time of each of the shape operations of sum, product, difference and symmetric difference on two shapes $a$ and $b$.

In the case of the algebra $U_1$, there exists a total, linear, order on the line segments within a co-equal class, such that, the comparison of two sets of co-equal segments reduces to a merging of both sets combined with a comparison of, consequently, consecutive segments in the resulting set. Given a resulting set $\{l_1, \ldots, l_n\}$, in order, we have that if $l_i$ and $l_j$ are disjoint with $i < j$, it follows that $l_i$ and $l_k$ are disjoint for all $k \geq j$. In this case, the shape operations of sum, product, difference and symmetric difference have a time complexity linear in the number of maximal segments $N$, i.e., $O(N f(n))$. The same bound holds for shapes of points in the algebra $U_0$. We show that, due to the choice of a segment as the basic case, a similar bound holds for shapes in $U_2$, whereas the bound is quadratic in the number of maximal segments for shapes in $U_3$. First, we determine the function $f$ for each of the algebras $U_k, 0 \leq k \leq 3$.

In $U_0$, two segments or points are co-equal if they are identical. Therefore, the operations combine, common and complement (applied to a pair of co-equal segments or points) have trivial definitions; two identical points contain one another, otherwise they are disjoint, but two points can never overlap nor share boundary. Thus, $f(n) = \Theta(1)$ for $U_0$. The resulting time complexities for the shape operations sum, product, difference and symmetric
difference, as well as for the shape relations contain, overlap, share boundary and disjoint, are $\Theta(N f(n)) = \Theta(N)$.

For $U_1$ (Krishnamurti, 1992a), the relations contain, overlap, share boundary and disjoint, when applied to a pair of co-equal segments, are functions of the boundary points of the line segments. The set of boundary points of the line segment(s) that results from combining two segments under the operations of combine, complement or common forms a subset of the (at most four) boundary points of both operands. Thus, $f(n) = \Theta(1)$ for $U_1$, also. The resulting time complexities for the shape operations of sum, product, difference and symmetric difference, as well as for the shape relations contain, overlap, share boundary and disjoint, are $\Theta(N f(n)) = \Theta(N)$.

For $U_2$, we have adopted a classification approach that classifies the boundary segments of either co-equal shape or segment into the classes of inner, outer, same-shared and opposite-shared segments with respect to the other shape. Then, the boundary of the shape resulting from the operation of sum (combine), product (common) or difference (complement), consists of the sum of the boundary segments from the appropriate classes. In Chapter 6 and Chapter 7, we have developed the corresponding algorithms for the procedures CLASSIFY (Algorithm 6.4) and CONSTRUCT (Algorithm 7.2). The former procedure classifies the boundary segments of both shapes in time $O((m + n) \log n)$ where $n$ equals the total size of the boundaries of both shapes and $m$ denotes the number of points of intersection between boundary segments from both shapes. The resulting classes have size $O(m + n)$. The latter procedure constructs the resulting shape from the given boundary in time $O(n' \log n')$ where $n'$ denotes the boundary size, i.e., $n' = O(m + n)$. The algorithms for the shape operations of sum, product and difference on co-equal shapes are given in Algorithm 5.1; their resulting time complexity is $O((m + n) \log n)$.

**Lemma 8.2** The shape operations of sum, product, difference and symmetric difference applied to two co-equal shapes $a$ and $b$ in $U_2$ have asymptotic time complexities linear in the number of maximal segments $N$ of both $a$ and $b$, i.e., $O(N f(n))$, where $n$ is a characteristic size of the boundaries of the maximal segments.
of a and b and the function f denotes the asymptotic time complexity of the shape operations and relations when applied to two co-equal segments.

Proof: Using Algorithm 5.1, the asymptotic time complexity of the operations of combine, common and complement on a pair of co-equal segments a and b is $O((m + n) \log n)$, where n is the number of boundary segments of a and b, $m = O(k n)$ is the number of points of intersection between boundary segments of a and boundary segments of b and k equals the number of co-equivalence classes of boundary segments of a and b. Thus, $f(n) = O((m + n) \log n)$. Given two co-equal shapes a and b consisting of $N$ maximal segments in total, for each segment i, let $n_i$ denote the number of boundary segments and let $m_i$ denote the number of intersection points between boundary segments of segment i and boundary segments of all other segments $j \neq i$. Then, the resulting time complexity for each of the shape operations of sum, product, difference and symmetric difference, as well as for the shape relations contain, overlap, share boundary and disjoint, is

$$\Theta(N f(n)) = \Theta(\sum_{i=1}^{N} f(n_i)) = O(\sum_{i=1}^{N} (m_i + n_i) \log n_i) = O(\sum_{i=1}^{N} (m_i + n_i) \log n)$$

$$= O((m_N + n_N) \log n_N)$$

with $n_N = \sum_{i=1}^{N} n_i$, the total number of boundary line segments and $m_N = \frac{1}{2} \sum_{i=1}^{N} m_i = O(k_N n_N)$, the total number of intersection points between boundary segments of a and boundary segments of b (and $k_N$ the total number of classes of boundary line segments).  

For $U_3$, we adopt the same classification approach with corresponding procedures CLASSIFY (Algorithm 6.7) and CONSTRUCT (Algorithm 7.5). The former procedure classifies the boundary plane segments of the two input shapes in time $O(k n \log n)$ where n equals the total size of the boundaries of both shapes and $k$ equals the number of co-equivalence classes of boundary segments from both shapes. The resulting classes have size $O(m + n)$, with $m = O(k n)$ the number of segments of intersection of boundary segments from both shapes. The latter procedure constructs the resulting shape from the given boundary in time $O(K m + k n' \log n')$ where $n'$ denotes the boundary size, i.e., $n' = O(m + n)$. However, this reduces to $O(K m + k n \log n)$ as the factor $k n'$ is on the same order as $k n$, that is, the number of line segments resulting from splitting the same k lines with respect to the original
boundary of size \( n \) or a resulting boundary of size \( n' \) composed of segments from the original boundary and split segments from those \( k \) lines. The algorithms for the shape operations of sum, product and difference are given in Algorithm 5.1; their resulting time complexity is \( \Omega(K_m + k n \log n) \), also.

**Lemma 8.3** The shape operations of sum, product, difference and symmetric-difference applied to two co-equal shapes \( a \) and \( b \) in \( U_3 \) have asymptotic time complexities quadratic in the number of maximal segments \( N \) of both \( a \) and \( b \), i.e., \( \Omega(N^2 f(n)) \), where \( n \) is a characteristic size of the boundaries of the maximal segments of \( a \) and \( b \) and the function \( f \) denotes the asymptotic time complexity of the shape operations and relations when applied to two co-equal segments.

*Proof:* Using Algorithm 5.1, the asymptotic time complexity of the operations of combine, common and complement on a pair of co-equal segments \( a \) and \( b \) is \( \Omega(K_m + k n \log n) \), where \( n \) is the number of boundary segments of \( a \) and \( b \), \( m = \Omega(k n) \) is the number of line segments of intersection between boundary segments of \( a \) and boundary segments of \( b \) and \( k \) equals the number of co-equivalence classes of boundary segments of \( a \) and \( b \). Thus, \( f(n) = \Omega(K_m + k n \log n) \). Given two co-equal shapes \( a \) and \( b \) consisting of \( N \) maximal segments in total, for each segment \( i \), let \( n_i \) denote the number of boundary segments and let \( m_i \) denote the number of line segments of points between boundary segments of segment \( i \) and boundary segments of all other segments \( j \neq i \). Let \( k \) denote a characteristic number of classes of boundary segments of a segment and let \( K \) denote a characteristic number of resulting simple boundaries of a segment. Then, the resulting time complexity for each of the shape operations of sum, product, difference and symmetric difference, as well as for the shape relations contain, overlap, share boundary and disjoint, is

\[
\Theta(N^2 f(n)) = \Theta(N \sum_{i=1}^{N} f(n_i)) = \Theta(N \sum_{i=1}^{N} K m_i + k n_i \log n_i) = \Theta(N (K m_N + k n_N \log n_N))
\]

\[
= \Theta(K_N m_N + k_N n_N \log n_N)
\]

with \( n_N = \sum n_i \), the total number of boundary line segments, \( m_N = \Theta(\frac{1}{2} \sum m_i) = \Theta(k_N n_N) \), the total number of line segments of intersection between boundary segments of \( a \) and
boundary segments of $b$, $k_N = \Theta(N k)$, the total number of classes of boundary line segments and $K_N = \Theta(N K)$, the total number of resulting simple boundaries.

Thus, we have shown that for each of the algebras $U_k$, $0 \leq k \leq 2$, the asymptotic time complexity for each of the shape operations and shape relations equals $\Theta(N f(n))$ where $f(n)$ is the time bound on the corresponding operations and relations applied to a pair of co-equal segments. However, this result for the algebra $U_2$ is due to the fact that we chose as basic element a segment, the boundary of which is actually composed of one or more simple boundaries. As such the boundary of a shape is no more complex than the boundary of a segment in the same algebra. For the algebra $U_3$, the relation equals $\Theta(N^2 f(n))$. These results are summarized in Table 8.1; $n$ ($n_N$) equals the total size of the boundaries of the input segments (respectively, shapes), $k$ ($k_N$) equals the number of classes of boundary segments, $k = \Theta(n)$, $m$ ($m_N$) equals the number of segments of intersection between boundary segments from both segments (or shapes), $m = \Theta(k n)$, and $K$ ($K_N$) equals the number of resulting simple boundaries, $K = \Theta(k)$. The lower bounds reflect the maximal representation adopted and, as such, do not correspond to lower bounds on the regularized set operations of union, intersection and difference on point sets or solids; the log factor is due to the ordering imposed by the canonical representation.

<table>
<thead>
<tr>
<th>$U_0$</th>
<th>$\Theta(1)$</th>
<th>$\Theta(N)$</th>
<th>$\Omega(N)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_1$</td>
<td>$\Theta(1)$</td>
<td>$\Theta(N)$</td>
<td>$\Omega(N)$</td>
</tr>
<tr>
<td>$U_2$</td>
<td>$\Theta((m + n) \log n)$</td>
<td>$\Theta((m_N + n_N) \log n_N)$</td>
<td>$\Omega((m_N + n_N) \log n_N)$</td>
</tr>
<tr>
<td>$U_3$</td>
<td>$\Theta((K m + k n) \log n)$</td>
<td>$\Theta((K_N m_N + k_N n_N) \log n_N)$</td>
<td>$\Omega((m_N + n_N) \log n_N)$</td>
</tr>
</tbody>
</table>

Table 8.1 The time complexities of the shape operations in the algebras $U_k$, $0 \leq k \leq 2$.
8.1.2 Comparison

For $U_2$, we compare the above results with similar results obtained by Nievergelt and Preparata (1982) and Mairson and Stolfi (1988). Nievergelt and Preparata present a plane-sweep algorithm that finds the regions defined by a self-intersecting polygon and runs in time $O((m+n) \log n)$ where $n$ denotes the number of vertices or edges of the polygon and $m$ denotes the number of intersection points. They present a second, similar, algorithm that computes all $m$ intersections of two convex maps (embedded planar graphs with convex regions) with a total of $n$ points in time $O(m + n \log n)$. They go on to suggest that the two algorithms may be combined to compute the regions of the intersection of two arbitrary maps (non-convex) in time $O((m+n) \log n)$. Mairson and Stolfi (1988) show that, given two sets $A$ and $B$ each consisting of non-intersecting line segments, such that $|A| + |B| = n$, time $O(m + n \log n)$ suffices to report all $m$ intersections of segments of $A$ with segments of $B$. This yields an algorithm to compute the intersection of two polygons with the same time complexity.

Consider the time complexity for the shape operations of sum, product and difference on two shapes of co-equal plane segments, in $U_2$. Let $n$ denote the total size of the boundaries of both shapes, that is, the number of boundary line segments of both shapes. Let $m$ denote the number of line segments of intersection between boundary segments of both shapes, $m = O(n^2)$. Then, the resulting time complexity equals $O((m + n) \log n)$. Thus, the asymptotic complexity of our algorithm is identical to Nievergelt and Preparata’s algorithm, but worse than Mairson and Stolfi’s algorithm. However, even if we use Mairson and Stolfi’s findings in order to improve our algorithm, the asymptotic complexity remains unaltered because we require a specific ordering on the boundary line segments which does not result from the algorithm as such.

For $U_3$, we compare the above results with similar results obtained by Karasick (1988) for a Star-Edge representation. Most algorithms for intersecting solids described in the literature pertain to convex polyhedra. Muller and Preparata (1978) describe an algorithm to intersect two convex polyhedra in time $O(n \log n)$, where $n$ equals the total number of vertices of both polyhedra. Their algorithm uses the concepts of duality between points and half-spaces and the convex hull of a set of points. The representation they develop is termed the doubly-
**Connected Edge List.** Hertel et al. (1984) use a different approach to determine the union, intersection or difference of two convex polyhedra, also using the *doubly-connected edge list* representation. They combine a space-sweep to find all intersection points of both polyhedra followed by a single boundary traversal and develop an algorithm that runs in \( O(n \log n) \) time, also. Other algorithms have been presented (Laidlaw et al., 1986; Paoluzzi et al., 1986; Segal and Séquin, 1988) but none is as complete (considering all degenerate cases) or as analyzed (regarding asymptotic complexity) as Karasick’s algorithm for the Star-Edge representation. Karasick (1988) includes a comparison of his algorithm with Laidlaw’s and Paoluzzi’s. Both are asymptotically slower than Karasick’s algorithm.

The *Star-Edge representation* (Karasick, 1988) is an explicit boundary representation with the following face properties:

1. Faces are regular sets with pairwise-disjoint interiors
2. Faces cover the surface of the represented solid
3. Faces are restricted to connected sets
4. Faces have a constant neighborhood

Note that a boundary representation is explicit if all incidences of features (vertices, edges and faces) are explicitly represented. This is in contrast to the maximal representation. The Star-Edge representation is “maximal” in the sense that a *Star-Edge face* is a maximal composition of connected, regular faces. However, the Star-Edge representation is not a canonical representation. Karasick develops an algorithm to determine the intersection of two solids \( A \) and \( B \) that assembles pieces of intersecting shells from \( A \) and \( B \) to obtain the resulting shells that are not entirely contained in the interior of either solid and, subsequently, adds the shells of \( A \) and \( B \) that are entirely contained in the interior of the other solid. This algorithm takes time \( O(K n^2 \log n) \), where \( n \) equals the total number of edges of \( A \) and \( B \) and \( K \) equals the number of (connected) shells in the boundary of the resulting solid.\(^3\)

Consider the time complexity for the shape operations sum, product and difference on two

---

\(^3\) Karasick specifies a time bound \( O(K D_A D_B \log (D_A D_B)) \), where \( D_A \) and \( D_B \) denote the numbers of directed edges in the Star-Edge representations of \( A \) and \( B \), respectively.
shapes, that is, the number of boundary plane segments of both shapes. We have that the number of boundary line segments of boundary plane segments equals $\Theta(n)$. Let $k$ denote the number of co-equivalence classes of boundary segments of both shapes, $k = O(n)$. Let $m$ denote the number of line segments of intersection between boundary segments of both shapes, $m = O(k n) = O(n^2)$. Let $K$ denote the number of simple boundaries for the resulting shape. Then, the resulting time complexity equals $O(K m + k n \log n)$. In order to compare both expressions of the time complexities involved, we need to compare the different steps in the algorithms, in particular, as well as take into account approximations that may have been used in the determination of such bounds, e.g., in one instance, the term $n^2$ in Karasick’s expression denotes the total number of edges involved, that is, corresponding to $m + n = O(k n)$.

Karasick’s algorithm for the regularized intersection of two solids $A$ and $B$, using a Star-Edge representation, consists of the following five steps:

1. Compute the cross-sections of solid $B$ corresponding to the carriers of the faces of $A$
2. Intersect the faces of solid $A$ with these cross-sections of solid $B$
3. Intersect the faces of solid $B$ with solid $A$
4. Construct the Star-Edge representation of $A \cap^* B$
5. Determine the connected components of $A \cap^* B$

These steps are achieved within the following time bounds. Let $D_A$ and $D_B$ denote the number of directed edges in the Star-Edge representations of solids $A$ and $B$, respectively. Let $F_A$ and $F_B$ denote the number of Star-Edge faces of solids $A$ and $B$, respectively. Step (1) takes time $O(D_B \log D_B)$ for each face of solid $A$, or, $O(F_A D_B \log D_B)$ in total; step (2) takes time $O(D_A D_B \log (D_A D_B))$ and step (3) takes time $O(D_A D_B + F_B D_A \log D_A)$. Combined, they result in an asymptotic time bound of $O(D_A D_B \log (D_A D_B))$, or, $O(n^2 \log n)$, where $n$ equals $D_A + D_B$. Step (4) takes time $O(D \log D)$, where $D$ is the number of directed edges in the Star-Edge representation of $A \cap^* B$. Whereas Karasick approximates this bound to $O(D_A D_B \log (D_A D_B))$ or $O(n^2 \log n)$, instead, using $m$ to denote the number of edges of intersection between faces of $A$ and faces of $B$, this can be written as $O((m + n) \log n)$. Step (5) takes time $O(K D \log D)$ or, similarly, $O(K (m + n) \log n)$, where $K$ denotes the number of
shells in the representation of $A \cap^* B$. Then, the global asymptotic time bound of Karasick’s algorithm is $O((Km + n^2) \log n)$.

This compares to $O(Km + kn \log n)$ for the shape operations in $U_3$. We notice two differences. Firstly, even though the Star-Edge representation recognizes the fact that co-planar faces can be grouped and treated as one, i.e., a Star-Edge face is a maximal composition of connected “faces”, it fails to use this characteristic to its fullest extend. Secondly, step (5) in the algorithm, which distinguishes “inner” and “outer” shells and determines the connected components of $A \cap^* B$, requires a sorting of intersection points among a line that intersects two “unexamined” shells. This results in a $\log n$ factor not present in the time bound for the corresponding algorithm for the shape operations in $U_3$.

### 8.2 Robustness

In the boundary representation of a geometric figure or shape, we distinguish geometric and topological information. The geometric information describes the embedding of the spatial elements in space; vertex coordinates and edge and face carrier equations constitute (part of) the geometric description. The topological description of a figure specifies the connectivity, i.e., adjacency relations, and orientation of faces, edges and vertices. In principle, a figure is completely specified by its geometry. However, a proper topological description is necessary to ensure a figure’s validity. The validity of figures in Euclidean space is generally ensured under the regularized Boolean operations.

Similarly, since an algebra of shapes is closed under the operations of sum, product and difference and the affine transformations, the validity of shapes is ensured under these operations. As such, the topological description of a shape is important only upon creation, and with respect to the algorithms for shape arithmetic. We have distinguished two major types of algorithms above: classification and construction algorithms. The classification algorithms, albeit assuming topological validity, use few of the topological information on shapes, and any topological description is lost in the classification process. The result of a classification is a set of classes of, yet, unrelated segments that were part of the boundary of the original shape(s). The construction algorithms are applied to a set of (a priori unrelated) segments from the operation-appropriate classes. In order to distinguish the boundary
cycles (polygons) or shells that define the resulting shape, a topology is constructed for the
boundary in the construction process.

The maximal element representation is a minimal representation in the sense that only the
least information necessary for efficient classification and construction algorithms is
maintained in the representation. As the connectivity information is irrelevant to the
classification process outlined in Chapter 6, only the orientation information is required in
the topological description. The geometric description includes both the (end)point
coordinates and the co-descriptors (i.e., carrier equations), because these co-descriptors are
necessary for a canonical representation. As a result, all boundary segments are regarded as
topologically separate structures even though their embedding in the Euclidean space
permits (as well as forces) geometric coincidence.

This distinction is particularly important from the standpoint of robustness. Whereas the
co-descriptor of any boundary element is unaltered by the algebraic operations, the implicit
topology may change, connecting boundary segments that were previously unrelated (and
vice versa). If some such elements coincide only inaccurately, the resulting shape is no
longer topologically valid and any subsequent classification or construction will result in an
incorrect interpretation and possible no reasonable result.

Accuracy and, leading to accuracy, precision, play a major role in any algorithmic
implementation. To ensure accuracy, one can rely on exact arithmetic, e.g., using integral or
rational coordinates and coefficients. A shape of points is considered rational if the points
are given by rational coordinates. In general, a shape is rational if its co-descriptor is
specified by rational coefficients and its boundary shape is rational. It can be proven that the
operations of sum, product, difference and symmetric difference on rational shapes have
rational results:4 The algebraic operations and their classification algorithms fundamentally
rely on algorithms for the intersection of plane and line segments. The point of intersection
of two (non-parallel) lines or line segments has rational coordinates if the co-descriptors of
both lines are rational (Krishnamurti and Earl, 1992). Also, the point of intersection of a

4. The algorithms described above have been implemented by the author using rational
arithmetic. Their correctness is shown by various applications (see Chapter 10).
the co-descriptors of both elements are rational; the point of intersection is the point of intersection of the line (or line segment) and any (rational) line in the plane. Similarly, the line of intersection of two (non-parallel) planes or plane segments has a rational co-descriptor if both planes have rational co-descriptors.\footnote{In fact, distinguished points (see Chapter 9) of a shape have rational coordinates if the co-descriptors of the shape elements are rational.}

As such, any algebra of rational shapes is closed under the algebraic operations of sum, product and difference, but not under the affine transformations. We illustrate this with a few rotations (see also Krishnamurti and Earl, 1992). In two dimensions, the pair of equations for a pure rotation about an imaginary Z-axis (under a right-handed coordinate system) is

\[
\begin{align*}
    x' &= \cos \theta \cdot x - \sin \theta \cdot y \\
    y' &= \sin \theta \cdot x + \cos \theta \cdot y
\end{align*}
\]

These equations are rational if \(a\) and \(b\) are rational, that is, if we can find integral values \(d\), \(e\) and \(f\) such that \(a = d / f\), \(b = e / f\) and \(d^2 + e^2 = f^2\). An example of such values is \(d = 3\), \(e = 4\) and \(f = 5\); the corresponding rotation angle \(\theta = \tan^{-1}(4 / 3) = 53.13\ldots^\circ\). Some rotations that are irrational in their pure form may be rationalized, in two dimensions, by an appropriate scaling: a 45° rotation is irrational without scaling (i.e., scaling factor 1) but becomes rational with a scaling factor \(\sqrt{2}\) (see Figure 8.1(a)). In three dimensions, however, an irrational rotation remains irrational irrespective of the chosen scaling factor (Figure 8.1(b)). Other rotations are always irrational, also in two dimensions, e.g., a 30° rotation (Figure 8.2).

Krishnamurti and Earl (1992) examine whether a transformation between rational shapes (i.e., one that maps one shape as a part of the other shape) is rational, that is, can be described by rational coefficients.

Rational shapes can be represented accurately in a computer using multiple precision integers. This guarantees robustness, but not without a cost. Sugihara and Iri (1989) shows that when intersecting three-dimensional, rectilinear structures, the number of significant digits more than quintuples. We refer to Hoffmann (1989b) for a more detailed examination of issues of robustness in geometric computations as well as approaches for developing
provable correct implementations of geometric algorithms. We also refer to Borodin and Munro (1975) for the complexity of multiprecision arithmetic.

Figure 8.1 (Ir)rational transformations: a composition of a 45° rotation and a $\sqrt{2}$ scaling is rational in two dimensions (a), but irrational in three dimensions (b).

Figure 8.2 A composition of a 30° rotation with an arbitrary scaling is always irrational, even in two dimensions.
Part III

Past, Present and Future
Chapter 9
Shape Emergence

Fundamental to the algebraic model is that, under the part relation, any part of a shape is a shape. That is, a shape defines an infinite set of shapes that are each part of the original shape. These shapes emerge under the part relation - albeit, they were not originally envisioned as such. The concept of emergent shapes is highly enticing to design search (Mitchell, 1993; Stiny, 1993).

Emergent shapes only become explicit when manipulated as such. Recognizing emergent shapes requires determining a transformation under which a specified similar shape is a part of the original shape. A shape rule constitutes a formal specification of shape recognition and subsequent manipulation. A shape rule has the form $lhs \rightarrow rhs$; the $lhs$ (left hand side) specifies the similar shape to be recognized, the $rhs$ (right hand side) of the shape rule specifies the manipulation leading to the resulting shape. Then, shape rule application consists of replacing the emergent shape corresponding to the $lhs$, under some allowable transformation, by the $rhs$, under the same transformation. A shape grammar combines a set of (semantically related) shape rules into a formal rewriting system for producing a language of shapes.
In this chapter, we introduce the shape grammars formalism, prove the irreversibility of shape rules and explore the theoretic aspects of shape recognition for shapes of elements from amongst points, line, plane and volume segments.

9.1 Shape Grammars

Grammars are formal devices for specifying “languages”; first used in logic by Post (1943), later by Chomsky (1957), who coined the term, in natural language processing, and more recently and originally, by Stiny (1980a) in design. Spatial grammars for generation and analysis have found wide ranging use in a variety of fields, e.g., tree grammars and graph grammars in syntactic pattern recognition (Fu, 1982), set grammars, shape grammars and parametric shape grammars in design and, in particular, architectural design (Stiny, 1977, 1980b; Knight, 1980; Koning and Eizenberg, 1981; Flemming, 1981, 1987; Carlson et al., 1991; Heisserman, 1991).

All grammars share certain definitions and characteristics. Grammars are defined over an algebra of objects, $U$, closed under the operations $+$ and $-$ and a set of transformations $F$. In other words, if $u$ and $v$ are members of $U$, then so are $u + f(v)$ and $u - f(v)$ where $f$ is a member of $F$. In addition, we can define a match relation $\leq$ on $U$ such that $f(u) \leq v$ whenever $u$ occurs in $v$, for some member $f$ of $F$.

In principle, there are four parts to a grammar $G = (N, T, R, I)$:

- nonterminal vocabulary $N$
- terminal vocabulary $T \subseteq U$
- finite set of rewriting rules (or productions) $R \subseteq \psi_1(T, N) \times \psi_2(T, N)$
- (set of) initial objects $I \subseteq \psi_1(T, N)$

$\psi_1$ and $\psi_2$ are set defining functions on $T$ and $N$. In general, the elements of the terminal and nonterminal vocabularies are members of $U$, though restrictions may apply.

A rewriting rule has the form $lhs \rightarrow rhs$. The $lhs$ (left hand side) of the rule contains elements from $T$ and $N$, but cannot be empty. The $rhs$ (right hand side) of the rule, in addition, may be empty. A rule applies to a particular object if the $lhs$ of the rule “matches” a part of the object under some allowable transformation $f$. Rule application consists of
replacing the matching part by the rhs of the rule under the same transformation. A rule \( a \rightarrow b \) is applicable to \( u \) whenever there is a transformation \( f \) such that \( f(a) \leq u \), in which case, under rule application, the object \( v \) is produced, given by the expression

\[
v = (u - f(a)) + f(b)
\]

We denote a rule application by \( u \Rightarrow v \) and say that \( u \) directly derives \( v \). A sequence of rule applications from object \( u \) to object \( v \) is denoted by \( u \Rightarrow^* v \); that is, \( u \Rightarrow \ldots \Rightarrow v \). In this case, we say that \( u \) derives \( v \).

The central problem in implementing grammars is the matching problem, that of determining the transformation \( f \) under which the relation \( \leq \) holds for \( f(a) \). Clearly, this problem depends upon on the representation of the elements of \( U \).

A grammar defines a language; that is, a set containing all objects generated by the grammar, where each generation starts with an initial object and uses rules to achieve an object that contains only terminals. A grammar is serial if a rule is applied to just one instance of a matching object, and parallel if a rule is applied to all such instances. Most grammars are used serially.

A uniform characterization for a variety of grammar systems is given in Gips and Stiny (1980). Krishnamurti and Stouffs (1993) survey a variety of spatial grammar formalisms from an implementation standpoint. A shape grammar is a formal rewriting system for producing languages of shapes (Stiny, 1980a). Shape grammars share many characteristics with other spatial grammars, yet unlike other spatial grammars, shape grammars operate directly on spatial forms, the algebra of which is the subject of this thesis. This chapter introduces the computationally challenging problem of subshape detection germane to shape rule application.

A shape can be augmented by distinguishing certain parts of the shape, which introduce additional spatial relations. The common definition of a shape grammar uses labeled points as non-terminals. Given a set \( L \) of symbols, which may be empty, we can define an algebra \( V_0 = U_0 \times \wp(L) \) of labeled points, where \( \wp(L) \) denotes the powerset of \( L \). Thus, a point is labeled if it has a set of symbols associated with it. A labeled shape is an element of the algebra \( V = U \times V_0 \). The algebra \( V \) has the same property as \( U \). A labeled shape is made up
of a shape and a finite, but possibly empty, set of labeled points. The set \((S, L)^+\) is the least set of all labeled shapes made up of shapes in the set \(S \subset U\) and labels in the set \(L\). If the empty shape is included, the least set is denoted as \((S, L)^*\).

A shape grammar \(G_s\) is defined as a four-tuple \(G_s = (S, L, R, I)\); \(S\) represents the terminal vocabulary; the set of symbols \(L\) specifies the nonterminal vocabulary; \(R\) is a finite set of shape rules in the form \(a \rightarrow b\), where \(a \in (S, L)^+\) and \(b \in (S, L)^*\); and \(I \in (S, L)^+\) is the initial shape. The vocabulary of \(G_s\) equals \((S, L)^*\).

A shape rule \(a \rightarrow b\) is a spatial relation between shapes \(a\) and \(b\); it applies to a shape \(s\) if we can find a transformation \(f\) such that \(f(a) \leq s\), in which case \(f(b)\) replaces \(f(a)\) in \(s\) under rule application. That is, when the shape rule is applied to the shape \(s\) it produces the shape, \(s' = f(a) + f(b)\). The set \(F\) of valid transformations, commonly considered, is the set of all Euclidean transformations\(^1\). However, since a shape algebra \(U\) is closed under the affine transformations, any subset of affine transformations will comply. A shape grammar \(G_s\) defines a language \(L(G_s)\), the set containing all shapes generated by the grammar \(G_s\) that have no symbols associated with them:

\[
L(G_s) = \{ x \mid x \in U \text{ such that } I \Rightarrow^* x \}
\]

Figure 9.1 shows a few shapes derived from an initial shape using only a single shape rule. All derivations are members of the language of the corresponding grammar.

### 9.1.1 Reversible and Irreversible Rules

A common characteristic of both string grammars and graph grammars is that their rules are reversible. That is, for any rule in such a grammar, a reverse rule can be constructed that, when applying, consecutively, the original and the reverse rule to any element of the algebra over which the grammar is defined, the result is identical to the original element. Furthermore, the reverse of a rule \(a \rightarrow b\) is the rule \(b \rightarrow a\).

---

1. By Euclidean transformations are meant the isometric transformations augmented with scale.
This is easily demonstrated for string grammars. A rule $a \rightarrow b$ applies to a string $u = xay$ and forms the string $v = xby$. The inverse rule $b \rightarrow a$ then applies to the string $v$ to form the original string $u$. A similar proof can be developed for graph grammars. However, this characteristic does not apply to either set grammars or shape grammars. A set grammar rule or shape rule may be either reversible or irreversible, depending on whether $rhs \leq lhs$ or not, respectively. We prove this for shape rules below.

Intuitively, we note that when combining two sets or shapes under the respective operation of “+”, identical elements “merge”. That is, only a single occurrence of the element appears in the resulting set or shape. Formally, in the case of set grammars, if $|u|$ denotes the cardinality of a set $u$, $|u + v| \leq |u| + |v|$. In the case of string or graph grammars, given the appropriate definition of the size of a graph, this would constitute strict equality. No comparable measure exists for shapes. The irreversibility of shape rules, in general, is proven below.
**Lemma 9.1** Given any shape rule $a \rightarrow b$ with $b$ not a part of $a$, no shape rule $x \rightarrow y$ exists such that, for any shape $u$, if $u \Rightarrow v$ under rule $a \rightarrow b$ and a transformation $f$, then $v \Rightarrow u$ under rule $x \rightarrow y$ and the same transformation $f$.

*Proof:* Assume that, given a rule $a \rightarrow b$ and a transformation $f$, there exists a rule $x \rightarrow y$ (which may be equal to $b \rightarrow a$) such that for any shape $u$ with $f(a) \leq u$, the shape that results from applying to $u$ the rules $a \rightarrow b$ and $x \rightarrow y$, in that order and under the same transformation $f$, equals $u$. Note that two shapes are considered equal if they are a part of each other. We may assume that the rule $x \rightarrow y$ applies under the same transformation; otherwise, we can always transform the rule such that it applies under the same transformation, without changing the rule application nor its scope. Thus, $u \Rightarrow v \Rightarrow w$ with $v = u - f(a) + f(b)$ and $w = v - f(x) + f(y) = u - f(a) + f(b) - f(x) + f(y) = u$.

Firstly, take $u \leftarrow f(a)$. Then, $u - f(a) = 0$ and $f(b) - f(x) + f(y) = f(a)$ or, identically, $b - x + y = a$. From the definition of sum, it follows that $y \leq a$ and $(b - x) \leq a$. Since $b \not\leq a$, it must be that $x \not\leq a$.

Secondly, take $u \leftarrow f(a) + f(b)$. Then, because sum and difference define a distributive lattice on $U$, $u - f(a) = (f(a) + f(b)) - f(a) = (f(a) - f(a)) + (f(b) - f(a)) = 0 + (f(b) - f(a)) = f(b) - f(a)$. From the definition of difference ($s - t \leq s$) it follows that $u - f(a) + f(b) = f(b) - f(a) + f(b) = f(b)$. Thus, $u = f(b) - f(x) + f(y) = f(a) + f(b) or b - x + y = a + b = b + a$. If we add $a$ to both sides of the equation we obtain $b - x + y + a = b + a + a or b - x + a = b + a (y \leq a)$, which is impossible, unless $x \leq a$. But we know from above that $x \not\leq a$.

Figure 9.2 illustrates an irreversible shape rule; Figure 9.3 illustrates a reversible shape rule.

**Lemma 9.2** A shape rule $a \rightarrow b$ is reversible if and only if $b \leq a$; the reverse rule is $b \rightarrow a$, or $f(b) \rightarrow f(a)$ for any transformation $f$.

*Proof:* We already know that a rule $a \rightarrow b$ is irreversible if $b \not\leq a$. It follows that, if $a \rightarrow b$ is reversible, $b$ must be a part of $a$. 
9.1 Shape Grammars

Figure 9.2 An example of an irreversible rule $a \rightarrow b$ (rule 1). We observe from the exemplar derivations that when applying the rules $a \rightarrow b$ and $b \rightarrow a$ (rule 2) subsequently and under the same transformation, the resulting shape may not equal the original shape.

Figure 9.3 An example of a reversible rule $a \rightarrow b$ (rule 1). We observe from the exemplar derivations that when applying the rules $a \rightarrow b$ and $b \rightarrow a$ (rule 2) subsequently and under the same transformation, the resulting shape equals the original shape.
The following proves the converse, that is, if \( b \leq a \) then \( a \to b \) is reversible and the reverse rule under the same transformation is \( b \to a \). Given the shape rules \( a \to b \) and \( b \to a \), and given any shape \( u \) with \( f(a) \leq u \), the shape that results from applying to \( u \) the rules \( a \to b \) and \( b \to a \), in that order and under the same transformation \( f \), is \( w = u - f(a) + f(b) - f(b) + f(a) \).

We have that \( b \leq a \) and, therefore, \( f(b) \leq f(a) \). Then, for any shape \( s \), \( s - f(b) + f(a) = s + f(a) \).

Thus, \( w = u - f(a) + f(b) - f(b) + f(a) = u - f(a) + f(b) + f(a) = u - f(a) + f(a) = u \). \( \square \)

Lemma 9.2 gives a sufficient condition for the reversibility of a shape rule independent of the shape to which it is applied. We can state a weaker condition for reversibility that is dependent on the shape under application.

**Lemma 9.3** A shape rule \( a \to b \) applied to a shape \( u \) under a transformation \( f \) is reversible if and only if \( f(b - a) \cdot u = 0 \).

**Proof:** Given the shape \( u \) and transformation \( f \), the shape rule \( a \to b \) and its inverse rule \( b \to a \), with \( f(a) \leq u \), the shape that results from the application of rules \( a \to b \) and \( b \to a \), in that order, is \( w = u - f(a) + f(b) - f(b) + f(a) \). We may assume that \( b \not\approx a \), and therefore, \( b = (b \cdot a) + (b \cdot a) = (b \cdot a) + (b \cdot a) \). Hence,

\[
w = u - f(a) + f(b) - f(b) + f(a) \\
= u - f(a) + f(b \cdot a) + f(b - a) - f(b - a) - f(b \cdot a) + f(a) \\
= u - f(a) + f(b \cdot a) + f(b - a) - f(b - a) + f(a) \quad \text{since } f(b \cdot a) \leq f(a) \\
= u - f(a) + f(b \cdot a) - f(b - a) + f(a) \quad \text{since } s + f(b - a) - f(b - a) = s - f(b - a) \text{ for any shape } s \\
= u - f(a) + f(b \cdot a) + f(a) - f(b - a) \quad \text{since } f(b - a) \cdot f(a) = 0 \\
= u - f(a) + f(a) - f(b - a) \quad \text{since } f(b \cdot a) \leq f(a) \\
= u + f(a) - f(b - a) \quad \text{since } f(a) \leq f(a) \\
= u - f(b - a) \quad \text{since for rule application } f(a) \leq u.
\]

Thus, \( w = u \) if and only if \( f(b - a) \cdot u = 0 \). \( \square \)

Irreversibility of shape rules distinguishes shape grammars from most other spatial formalisms. The significance from a design standpoint, to us, at any rate, stems from the
fact that designing is a temporal activity. The irreversibility of a rule has the effect of time stamping each rule application and, thus, capturing design “intent” at any given time.

9.2 Subshape Detection

Subshape detection relates to finding one or all valid transformations under which the \textit{lhs} of a shape rule is a part of a given shape. A solution to this problem consists of finding a correspondence between the spatial elements in the \textit{lhs} and elements of the given shape, and determining the transformation that represents this correspondence. The equivalent problem for set or graph grammars, termed subset and subgraph detection, consists of searching for either a single entity or a group of entities within a set or a graph. Such a search is straightforward, i.e., it requires a one-to-one matching of entities that are identical under a certain transformation, though not always computationally efficient, e.g., the subgraph isomorphism problem is \textit{NP}-complete (Garey and Johnson, 1979). On the other hand, a prime requirement for shape grammar computation is that any subshape of a shape is spatially replaceable. That is, a shape, even though with definite description, has indefinitely many “touchable” parts; a shape is an \textit{individual} \(^2\) (Stiny, 1982) and this is reflected in the part relation defined on shapes.

For \(U_0\) and \(U_1\) \textit{distinguishable points} serve as the basis for reducing the problem (Krishnamurti, 1981; Chase, 1989; Krishnamurti and Earl, 1992). In general, for a given shape, a distinguishable point corresponds to any point, or labeled point, that is a part of the shape, as well as any point of intersection of two or more carriers of segments in the shape. These points are distinguishable by the fact that they retain their properties under the part relation and any affine transformation. The application of this concept to \(U_k\), for \(k > 1\), is explored below. In \(n\) dimensions, a correspondence between \(n+1\), not co-hyperplanar, distinguishable points uniquely determines an affine transformation. However, \(n\) such points suffice to achieve a determinate number of valid, Euclidean, transformations, if the corresponding “point figures” are similar. Any such transformation found remains subject

\(^2\) The concept of individuals differs from the generally accepted concept of classes (or sets) in that no subdivision into subclasses or members is established or suggested \textit{a priori} (Leonard and Goodman, 1940).
to an evaluation with respect to the entire \( \text{lhs} \). Otherwise, if \( n \), not co-hyperplanar, distinguishable points cannot be determined in the \( \text{lhs} \), then, an indeterminate number of valid transformations may exist. In such cases, one can isolate a set of “base” transformations from which the other transformations can be generated.

In the sequel, we restrict the discussion to Euclidean transformations. Note that even in the case of shapes with rational descriptions, the resulting transformations may not be rational (Krishnamurti and Earl, 1992). As such, the constructions described below do not guarantee exact arithmetic.

No algorithm for subshape detection is given. Krishnamurti (1981) specifies a subshape detection algorithm for shapes of points and line segments in two dimensions. A similar algorithm is expected for shapes of points, line, plane and volume segments in three dimensions.

9.2.1 Shape Recognition in \( U_{n,3} \)

We give an overview of the determinate and indeterminate cases for each of the algebras \( U_{n,3} \) \((n \leq 3)\). Since the representation of a shape in \( U_{n,3} \) lies ultimately in \( E^3 \), we distinguish the cases in \( E^3 \) using as primary distinguishable elements the distinct carriers of the segments in the \( \text{lhs} \). Additional distinguishable points may be constructed for the purpose of generating a determinate number of valid transformations, yet the cases are distinguished solely on the primary distinguishable elements. The cases below are grouped by each algebra and numbered accordingly: the first digit denotes the dimensionality of the algebra (i.e., the segments of the algebra).

The cases for \( U_{0,3} \) are trivial. There exists one determinate case, when at least three non-colinear (distinguishable) points in the given shape can be found. Otherwise, a possible indeterminate number of valid transformations between the shapes exist.

**Case 0.1** There are three non-colinear points.

Three non-colinear points uniquely determine an affine transformation in \( E^2 \), i.e., for the plane they define. In \( E^3 \), if the “point figures” are similar, this results in two Euclidean transformations, of which one can be derived from the other using a reflection in this plane.
A fourth, non-coplanar, point suffices to invalidate one of these transformations. Krishnamurti and Earl (1992) describe the computation of both transformations.

**Case 0.2** There are two (distinct) points.

An indeterminate number of valid transformations exist. Given a base transformation that maps both points in the *lhs* onto the respective points in the given shape, the full set of possible transformations is derived using rotations about the line connecting both points and a reflection in a plane through this line.

**Case 0.3** There is a single point.

An indeterminate number of valid transformations exist. These can be derived from a base transformation using (three-axes) rotations about this point, scalings that leave the point fixed and a reflection in a plane through this point.

Krishnamurti and Earl (1992) fully describe the possible cases for $U_{1,3}$ (as well as $V_{1,3}$). There are two determinate cases, these are illustrated in Figure 9.4.

**Case 1.1** There are two skew lines.

Skew lines are not parallel, nor do they intersect in a point. The common perpendicular of two skew lines defines two distinguishable points, i.e., the feet of this perpendicular on the lines. Since two points are sufficient to determine a fixed scaling factor, an additional
(distinguishable) point may be constructed on one of the lines. As such, this problem reduces to Case 0.1, with three distinguishable points, for which a determinate number of possible valid transformations exist. Krishnamurti and Earl (1992) describe a variant to this construction.

**Case 1.2**  
There are three coplanar lines, not all parallel and not all concurrent at a single point.

If no two lines are parallel, then, the intersection points of these lines constitute three distinguishable points. Otherwise, the two points of intersection constitute two distinguishable points that can be augmented with a third point constructed similar to Case 1.1.

There are three possible indeterminate cases as illustrated in Figure 9.5.

**Case 1.3**  
There are two non-parallel lines.

From a base transformation, all other transformations result from scalings that leave the point of intersection fixed and a reflection in the plane defined by both lines.

**Case 1.4**  
There are two (parallel) lines.

The full set of transformations is generated by composing a base transformation with translations along the direction of the lines together with a reflection in a plane perpendicular to the parallel lines.

---

**Figure 9.5** Examples illustrating the indeterminate cases for \( U_{1,3} \): (a) Case 1.3, (b) Case 1.4 and (c) Case 1.5.
Case 1.5 There is a single line.

The full set of possible transformations is found by composing the base transformation with rotations about the line, denoted as $l$, scalings that leave a point on $l$ fixed, translations along the direction of $l$, and a reflection in a plane normal to $l$.

Krishnamurti and Stouffs (1993) distinguish the following cases in $U_{2,3}$. There is a single determinate case in $U_{2,3}$, illustrated in Figure 9.6.

Case 2.1 There are four planes, not all parallel and not all lines of intersection are parallel, concurrent or coincide.

We can find two skew lines of intersection and reduce the problem to Case 1.1 consequently, for which there exists a determinate number of valid transformations.

There are five indeterminate cases in $U_{2,3}$, illustrated in Figure 9.7.

Case 2.2 There are three planes, and not all the lines of intersection are parallel or coincide.

Another way to formulate this is the following: There are three planes, and their normal vectors are linearly independent. All the lines of intersection are concurrent at a single
point, and the problem reduces to Case 1.3, for which there exists a possible indeterminate number of valid transformations (under scaling and reflection).

**Case 2.3** *There are three planes and they do not intersect in a single line.*

All the lines of intersection may be parallel, but they do not coincide. This problem reduces to Case 1.4, for which there exists a possible indeterminate number of valid transformations (under translation and reflection).

**Case 2.4** *There are two non-parallel planes.*

The full set of transformations is generated by composing a base transformation with translations along the direction of the line of intersection together with scalings that leave a point on this line fixed and a reflection in a perpendicular plane.

**Case 2.5** *There are two (parallel) planes.*

A base transformation maps the carriers of the two planes in the lhs onto the respective carriers of two planes in the given shape. All other transformations result from translations
along two perpendicular axes parallel to these planes, rotations about a line normal to both planes and a reflection in a plane through this line.

**Case 2.6** *There is a single plane.*

A base transformation maps the carrier of a plane in the *lhs* onto the carrier of a plane in the given shape. The full set of possible transformations is generated by composing this base transformation with translations along two perpendicular axes parallel to the plane, scalings that leave a point on the plane fixed, rotations about a line normal to the plane and a reflection in a plane through this line.

Finally, in $U_{3,3}$, all shapes are necessarily co-hyperplanar and no distinguishable points, nor distinguishable lines or planes, can be constructed. As a result, an indeterminate number of transformations exist, the base transformation of which is the identity transformation.\(^3\)

Figure 9.8 shows the base transformations for two shapes of plane segments (Case 2.2).

---

\(^3\) This generalizes to all algebras $U_{n,m}$ except for $U_{0,0}$ where only one transformation, namely, the identity transformation, exists.
9.2.2 Shape Recognition Revisited

The preceding discussion considers all the possible cases for each of the algebras $U_{n,3}$, $n \leq 3$. However, in many cases a shape grammar is defined over a cartesian product of algebra’s. In Section 9.1 we define a shape grammar over an algebra $V = U \times V_0$ of labeled shapes, where a labeled shape is made up of a shape and a finite set of labeled points. More generally, a shape grammar can be defined over any cartesian product of algebras $U_i \times U_j \times \ldots$. As such, it is insufficient to enumerate the cases only for each of the algebras $U_{n,3}$ separately. Following, we consider an enumeration for any cartesian product of algebras from $U_{0,3}$ through $U_{3,3}$.

Consider a shape $s$ in the algebra $U_0 \times U_1 \times U_2 \times U_3$. The primary distinguishable elements can be augmented with constructed ones, such as the point of intersection of two lines, the line of intersection of two planes, the normal line to a plane through a point, and so on. Based on the combinations of independent transformations that yields the full set of possible valid transformations from a base transformation, only ten primary cases remain, of which one is determinate and the remaining nine indeterminate. We consider the degrees of freedom (DOF) corresponding to an (possible) indeterminate case to be the number of independent transformations as such (not including a possible reflection). Single transformations include a rotation about a line, a translation along a line and a scaling. A single plane defines two independent translations; a general rotation about a single point constitutes three independent rotations. We use $T$ to denote a translation, with $T_l$ a translation along a line $l$ and $T_u$ and $T_v$ two (perpendicular) translations parallel to a plane. We use $R$ to denote a rotation, with $R_l$ a rotation about a line $l$, $R_n$ a rotation about the normal $n$ to a plane and $R_x$, $R_y$ and $R_z$ rotations about the major axes. We use $S$ to denote a scaling.

The ten determinate and indeterminate cases are summarized in Table 9.1. Cases II through IV correspond to a single DOF, either rotational, translational or scaling. Cases V and VI have two DOF of which one corresponds to a scaling necessarily, the other is either rotational or translational. Cases VII and VIII have three degrees of freedom: a single distinguishable line, results in a translational DOF along the line and a rotational DOF about
the line, as well as a scaling; two parallel planes give way to a single rotational DOF and two translational DOF. Cases IX and X have a maximal four degrees of freedom: either three rotational or one rotational and two translational, together with a scaling. No other combinations are possible: an axis of rotation corresponds to a single DOF; a center of rotation to three DOF, no construct allows for only two rotational DOF. Similarly, any distinguishable element removes at least one translational DOF; only a distinguishable plane allows for two translational DOF, but only a single rotational DOF.

Each primary case defines a set of secondary cases, each of which can be reduced to the primary case by constructing additional distinguishable elements. However, it is computationally expensive to construct all possible distinguishable elements a priori, in
order to determine the particular primary case. Therefore, we list below the secondary cases for each primary case, for all possible cases of combinations of primary distinguishable elements. Table 9.2 summarizes all non-redundant cases with at least one distinguishable point. Table 9.3 summarizes all non-redundant cases without distinguishable points but at least one distinguishable line. Table 9.4 summarizes all cases with only distinguishable points.

<table>
<thead>
<tr>
<th>Points</th>
<th>Lines</th>
<th>Planes</th>
<th>Case</th>
<th>DOF</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 • non-colinear</td>
<td></td>
<td></td>
<td>La</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1 • non-perpendicular</td>
<td></td>
<td>Lb</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>II.a</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1 • non-colinear</td>
<td></td>
<td>Lc</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1 • colinear</td>
<td>1 • non-coplanar • non-perpendicular</td>
<td>Ld</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1 • colinear</td>
<td>1 • coplanar</td>
<td>III.a</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1 • colinear</td>
<td></td>
<td>IV.a</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2 • non-colinear intersection</td>
<td></td>
<td>Le</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2 • colinear intersection</td>
<td></td>
<td>III.b</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1 • non-coplanar</td>
<td></td>
<td>II.b</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1 • coplanar</td>
<td></td>
<td>V.b</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>IX.a</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 9.2 All non-redundant cases with at least one distinguishable point, based on combinations of primary distinguishable elements.

4. An example of redundancy would be two distinguishable points and one line non-colinear with at least one point: the second, possibly colinear, point is redundant as Case I.c shows.
9.2 Subshape Detection

planes, these correspond to the cases for $U_2, 3$. The cases are numbered according to the primary case.

Case I.a (primary case I) is identical to Case 0.1; it applies towards Case 1.1, Case 1.2 and Case 2.1 as well.

<table>
<thead>
<tr>
<th>Distinguishable elements</th>
<th>Lines</th>
<th>Planes</th>
<th>Case</th>
<th>DOF</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 • coplanar • not all parallel • non-concurrent</td>
<td>2 • skew</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 • coplanar • non-parallel</td>
<td></td>
<td>1 • non-concurrent</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 • coplanar • non-parallel</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 • parallel</td>
<td></td>
<td>1 • non-parallel</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 • parallel</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2 • non-concurrent • not both parallel or perpendicular to line</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2 • perpendicular to line</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1 • non-parallel • non-perpendicular</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1 • parallel</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1 • perpendicular</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1 • coplanar</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 9.3 All non-redundant cases without distinguishable points but at least one distinguishable line, based on combinations of primary distinguishable elements.
There are three non-colinear points. We reduce each of the following cases to Case I.a by constructing the necessary distinguishable points.

**Case I.b** There are two points together with a plane not perpendicular to the line defined by both points.

If at least one point is not coincident with the plane, then construct the foot of the perpendicular from this point onto the plane as a third distinguishable point. These three points are not colinear, otherwise the plane would be perpendicular to the line through both points. If both points are coincident with the plane, then construct a point on the normal with the plane through one point, such that the distance to this point is identical to the distance between the two original points (Figure 9.9).

**Case I.c** There is a point and a line not colinear with the point.

Construct the foot of the perpendicular from the point onto the line and construct a third point on the line at equal distance from the foot as the distance between the foot and the first point (Figure 9.10(a); Krishnamurti and Earl, 1992).

---

### Table 9.4 All non-redundant cases with only distinguishable planes

<table>
<thead>
<tr>
<th>Distinguishable Planes</th>
<th>Case</th>
<th>DOF</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 • not all parallel • not all intersection lines parallel or concurrent</td>
<td>I.k</td>
<td>0</td>
</tr>
<tr>
<td>3 • not all intersection lines parallel</td>
<td>III.e</td>
<td>1</td>
</tr>
<tr>
<td>3 • not concurrent in a single line</td>
<td>IV.e</td>
<td>1</td>
</tr>
<tr>
<td>2 • non-parallel</td>
<td>VI.b</td>
<td>2</td>
</tr>
<tr>
<td>2 • parallel</td>
<td>VIII.a</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>X.a</td>
<td>4</td>
</tr>
</tbody>
</table>
Case I.d There is a point, a line colinear with the point, and a plane not coincident with the point nor perpendicular to the line.

Construct the foot of the perpendicular from the point onto the plane. Since this point is not colinear with the line, we have a situation similar to Case I.c. However, we can construct a third point on the line at an equal distance from the original point as the distance from the foot to this point (Figure 9.10(b)).

Case I.e There is a point together with two planes which are not both coincident with the point.
Since the line of intersection is not colinear with the point, this problem is reduced to Case I.c.

**Case I.f** *There are three coplanar lines, not all parallel and not all concurrent at a single point.*

See Case 1.2.

**Case I.g** *There are two skewed lines.*

See Case 1.1.

**Case I.h** *There are two coplanar, non-parallel lines together with a plane, not all concurrent at a single point.*

Consider the point of intersection of both lines together with one of the lines, that is not perpendicular to the plane, and the plane; this corresponds to Case I.d.

**Case I.i** *There are two (parallel) lines together with a plane that is not parallel to the lines, nor coincides with a line.*

Construct the intersection points of both lines with the plane. Construct a third point on one of the lines at equal distance from the point of intersection of this line with the plane as the distance between both intersection points (Figure 9.11(a)).
Case I.j  There is a single line together with two planes, not both parallel or perpendicular to the line, such that they are not all concurrent at a single point.

The construction is dependent on whether one of the planes is parallel to the line, or not. If one plane is parallel to, but not coincident with the line, while the other plane is neither parallel to, nor coincident with the line, then, the line of intersection of both planes is skew with respect to the original line (Figure 9.11(b)). This reduces to Case I.g or Case 1.1. Otherwise, neither plane is parallel to, nor coincident with the line, while not both planes are perpendicular to the line and all three elements are not concurrent at a single point. Consider the (two) points of intersection of the line with both planes, together with the foot of the perpendicular from one of the intersection points onto the other plane. These three points cannot be collinear (Figure 9.12).

Case I.k  There are four planes, not all parallel and not all lines of intersection are parallel, concurrent or coincide.

See Case 2.1.

Case II.a (primary case II) specifies a single axis of rotation (without scaling); it is identical to Case 0.2.

Case II.a  There are two (distinct) points.
We reduce each of the following cases to Case II.a by constructing the necessary distinguishable points.

**Case II.b** There is one point and one plane not coplanar with the point.

Construct the foot of the perpendicular from the point onto the plane. The resulting axis of rotation (defined by both points) is perpendicular to the plane, such that the plane is mapped onto itself under the transformation.

**Case II.c** There is one line and two planes perpendicular to the line.

The two points of intersection of the line with the planes are distinct.

Cases III.a through III.e specify a scaling. Case III.c is the representative case (primary case III); it is identical to Case 1.3 and applies also towards Case 2.2. We reduce each of the following cases to Case III.c by constructing the necessary distinguishable lines.

**Case III.a** There is one point, one line and one plane, all coincident.

Construct a second line perpendicular to the first line, coincident with both the point and the plane.

**Case III.b** There is a point together with two planes coincident with the point.

Construct the foot of the perpendicular from the point onto the plane. The resulting axis of rotation (defined by both points) is perpendicular to the plane, such that the plane is mapped onto itself under the transformation.

**Case III.c** There are two non-parallel lines.

See Case 1.3.

**Case III.d** There is one line and one plane; these are not parallel nor perpendicular.

Construct the line of intersection of a second plane coincident with the line and perpendicular to the first plane, with the first plane.
Case III.e  There are three planes, and not all the lines of intersection are parallel or coincide.

See Case 2.2.

Case IV.a (primary case IV) specifies a single direction of translation (without scaling); it is identical to Case 1.4 and applies also towards Case 2.3.

Case IV.a  There are two parallel lines.

We reduce each of the following cases to Case IV.a by constructing the necessary distinguishable lines.

Case IV.b  There is one line and one parallel plane.

There exists a unique second line coincident with the plane and parallel to the first line, such that the foot of the perpendicular from any point on the first line lies on the second line (i.e., the second line constitutes a normal projection of the first line onto the plane).

Case IV.c  There are three planes and they do not intersect in a single line.

See Case 2.3.

Case V.a (primary case V) specifies a single axis of rotation with scaling.

Case V.a  There is a single point and a line coincident with the point.

The line constitutes the axis of rotation, the single point inhibits any translation but allows for a scaling (Krishnamurti and Earl, 1992). We reduce each of the following cases to Case V.a by constructing the necessary distinguishable line and/or point.

Case V.b  There is a single point and a plane coincident with the point.

The normal line to the plane that is coincident with the point, constitutes the axis of rotation.

Case V.c  There is a single line and a plane perpendicular to the line.
The line constitutes the axis of rotation, the point of intersection of the line and plane constitutes the fixed point.

Case VI.a (primary case VI) specifies a single direction of translation with scaling.

\textbf{Case VI.a} \textit{There is a single line and a plane coincident with the line.}

The line defines the direction of translation, the plane inhibits any rotation about the line but allows for scaling.

\textbf{Case VI.b} \textit{There are two non-parallel planes.}

See Case 2.4.

Case VII.a (primary case VII) specifies a single axis of rotation, a single direction of translation and a scaling; it is identical to Case 1.5.

\textbf{Case VII.a} \textit{There is a single line.}

Case VIII.a (primary case VIII) specifies a single axis of rotation and two (perpendicular) directions of translation (without scaling); it is identical to Case 2.5.

\textbf{Case VIII.a} \textit{There are two (parallel) planes.}

Case IX.a (primary case IX) specifies three (perpendicular) axes of rotation and a scaling; it is identical to Case 0.3.

\textbf{Case IX.a} \textit{There is a single point.}

Case X.a (primary case X) specifies a single axis of rotation, two (perpendicular) directions of translation and a scaling; it is identical to Case 2.6.
9.2 Subshape Detection

Case X.a  *There is a single plane.*

This completes the enumeration of the secondary cases.

In a similar vein, one can describe a treatment of labeled shapes, where elements such as lines and planes can be associated with labels. This is not done here.

9.2.3 Variations on a Common Theme

The previous classification, according to the number and type (i.e., whether rotational, translational, or scaling) of degrees of freedom, is based solely on distinguishable elements. That is, only the carriers of the shape segments and not the segments themselves define the specific classification. Each class is characterized by one or more base transformations, from which all possible transformations can be derived in correspondence to the DOF of this class. A possible transformation is a transformation that maps the given shapes under the part relation. As such, the part relation defines a constraint on the possible transformations. This constraint is particular to subshape recognition but not necessarily to shape emergence.

Gero and Yan (1993) extend the notion of emergent shapes to *emergent visual shapes*, i.e., emergent shapes that are cognitively recognized as such. In particular, using shapes of line segments, they consider as an emergent visual shape, any shape that is embedded in the carriers of the given shape and the segments of which are bounded by the points of intersection of the carriers. These carriers and their points of intersection exactly define the distinguishable elements used to determine the class of transformations under shape recognition. The resulting set of possible transformations is constrained by the embedding of the segments of the transformed shape in the carriers (instead of the segments) of the given shape.

In parametric shape grammars (Stiny, 1977, 1980a), shape recognition is dependent on a set of parameters specified on the emergent shape, together with constraints and bounds on the parameters. Gero and Yan (1993) specify a symbolic representation for emergent shapes...
using only topological, geometric and dimensional constraints. A theory that combines these approaches as well as allows for general types of constraints is missing still.
The results of this work have so far been applied to two separate applications. The first one concerned the use of robots in building construction simulation. The maximal element representation is adopted to represent the geometry of the building components and the building under construction, as well as the geometry of the robots in the simulation. The operations of product and/or intersection serve to detect and avoid possible collision among robots and between robots and the building or other obstacles, when planning a robot motion path. The second application explores the challenges and the implications that the algebraic model for shapes poses to computational design in an interactive and generative design environment.

10.1 RUBICON

The first application, RUBICON, is a rule-based simulation program developed for application to building construction, specifically concerning the use of robots in building construction simulation. The purpose of this work was to examine automated building
construction from the stand-point of construction management, site characteristics, man-
robot cooperation and robot diversity. We developed a general representation for robots in
building construction, to simulate the robots’ capabilities to operate within different building
projects and in cooperation with human labor crews. We introduced motion rules as a way
to capture the diversity in the different robot types in a uniform manner. Motion rules are
used as a tool to embody the discretization of time in the simulation; they serve as a way to
specify the motional capabilities of a manipulator in terms of a set of control parameters, the
latter representing the degrees of freedom of the manipulator; and they supply a way to solve
the discrepancy that may occur between specifying a robot’s configuration on one hand, and
controlling its motion on the other hand. We developed a motion language to model the
behavior of robots in the real-world environment of building construction. The motion
language builds upon the motion rules, and is used to specify the actions and operations of a
manipulator when faced with different tasks and/or situations. The motion language, when
combined with a simple collision avoidance algorithm for path planning that is based on a
‘generate and test’ procedure, allows for the simulation of robots in building construction.
An overview of the RUBICON project is given below (Stouffs et al., 1993a, 1993b),
followed by a discussion on the contributions of the representation scheme for shapes to the
RUBICON program.

10.1.1 The RUBICON Project

Robots in building construction include any form or type of automated mechanical
manipulator characterized by spatial motion, that is employed in the construction process.
Robots in building construction may vary greatly in size, mobility, purpose (the tasks or
operations they are intended for) and adaptability to different situations within a real-world
environment. In particular, we consider robots for construction that are characterized by
their ability to lift, move and place sizable construction components from the delivery
location to the placement location. As examples, we consider three different robot types: a
robot crane for handling heavy components (Figure 10.1), a robot towmotor for palettized
materials (Figure 10.2), and an exterior wall finishing robot (Figure 10.3).

1. The RUBICON project was funded, in part, by the Japan Research Institute.
The description of a robot encompasses a representation of its configuration, i.e., position and orientation, over time, its motional capabilities and its configurational limits. We use a fixed number of frames and transforms to describe a robot’s configuration at a particular time, introduce a set of control parameters to capture possible configurational changes over time, associate motion rules to these control parameters to describe a robot’s motional
A robot can be described in terms of the configuration of its components to each of which we rigidly attach a coordinate system or frame (Craig, 1989; see Figure 10.4). A transform describes one frame relative to another. A minimal description of a robot involves the following transforms: $U_B T$ describes the base frame relative to the universal frame; $B_T T$ describes the tool frame with respect to the base frame; and $T_O T$ describes the object frame.
with respect to the tool frame. The object frame is described relative to the universal frame by the composite transform $U_T = T T T$.

The ability of these transforms to change over time and, as a result, the relative configuration of the bodies they describe, is captured in a set of control parameters. For each transform, these parameters correspond to the respective degrees of freedom (DOF) of the body relative to its reference frame. Physically, the relationship between the end-effector and the base of the robot is defined by a set of links and joints, generally formed into an open kinematic chain. These joints can be either rotational (change of orientation) or prismatic (translational change). The number of DOF of the robot’s end-effector, with respect to the base and for an open kinematic chain, corresponds in most cases to the number of joints. The set of valid control parameters consists of $x$, $y$, $z$, $\psi$, $\phi$, and $\theta$, where $x$, $y$, $z$ define translations parallel to the respective axes and $\psi$, $\phi$, $\theta$ define rotations about the $X$-, $Y$- and $Z$-axis, respectively.

In the case of a mobile robot, the motional capabilities of the base of the robot often are dependent on the current configuration of the base and, thus, defined locally. However, the configuration itself is defined relative to a reference frame which is fixed with respect to the body’s motion, and is thus defined globally. Motion rules are conceived to describe a robot’s motional capabilities possibly independent of its configuration. Foremost, however, they form a tool to embody the discretization of time in the simulation process. A motion rule generally defines the value of a control parameter at time $t + \Delta t$, where $\Delta t$ denotes the time step, in terms of the control parameter’s values at time $t$ and their initial and step values. The initial values are the values of the control parameters at time $0$. Under the assumption of constant velocity, for each control parameter we can specify a motion step as a constant value. Each parameter also has a minimum and maximum value specified in accordance with the relative workspace of the body. This workspace is roughly the volume of space which the body can reach in at least one orientation, relative to its reference frame (Craig 1989).

In the simulation, we are concerned with the role that a particular robot plays in the construction process and with its place in and its relation to the environment. That is, we are
interested in the interactions of the robot with the site, the building under construction, the human labor crews, and other robots involved in the construction. This relation and, therefore, the robot’s behavior, is constrained by physical obstacles, task-specified and other interactions, and safety and other considerations. A motion language is developed that serves as a means to specify this behavior and, in particular, to specify the order in which the allowed motion steps should be proposed, for each robot or body (Stouffs et al., 1993c). The choice of a specific motion step is dependent on the particular situation of the robot at the moment, according to the specified behavior. When a motion step has been chosen, all motion rules that contain this motion step are invoked, and the corresponding control parameters are updated.

Such a motion scheme fits easily within the framework of a local path planning method. In path planning we distinguish between global and local techniques. Global techniques are generally based on the concept of a configuration space (C-space) introduced by Lozano-Pérez and Wesley (1979). The disadvantages of this approach are due to the fact that C-space has high dimensionality for bodies with many DOF, and the technique is computationally expensive. It is also inefficient when dealing with a dynamically changing environment, as is the case for building construction. Local strategies may function both in task space and in configuration space. They are generally based on artificial potential functions (Khatib 1986), but exhibit a limitation that is due to the appearance of local minima.

In order to deal with a dynamically changing environment it is often useful to follow a local approach where obstacles are avoided as they are encountered (Lumelsky 1986). In this motion planning approach a path is built up from single motion steps in a bottom-up fashion. Such a technique may be either goal-driven or path-driven. Lozano-Pérez and Wesley (1979) present a simple collision avoidance algorithm that is based on the ‘generate and test’ paradigm. Adapted to motion planning, the algorithm takes the form:

1. compute the volume swept out by the moving object for the proposed motion step;
2. determine the intersection of the swept volume and the surrounding obstacles;
3. if an intersection exists, propose a new motion step.
The RUBICON program implements these findings. From a specified task schedule, the program generates and simulates a motion path for each robot action, avoiding obstacles and incorporating interaction, safety and other considerations. The program can be used to study different task plans, report on different robot types, study alternate robot-human crew mix examples, and produce measurements in terms of time or cost. Working examples include an automated crane hoist and an automated towmotor active in the construction of a typical precast concrete residential building (Figure 10.5).

The simulation takes a building construction task plan as input. This is a detailed plan describing the construction elements and the construction process as a task schedule. A task is to be performed either by a human crew, or by a robot. The simulator then translates each task into a robot motion plan, i.e., a sequence of motion steps, using a rule-based description of the robot agents. The motion plans reflect the respective robot’s motional capabilities and limits, and avoid any collision. The output of the RUBICON program is a graphical simulation of the construction process as specified in the task plan, with a visualization of the motional actions of the robot agents and of the transportation of the construction
components by these agents, and with a specification of the process time. Figure 10.6 illustrates the RUBICON graphical user interface.

The RUBICON program serves the project planning engineer in studying alternate task plans, alternate resource mixes and the use of alternate robot types in the construction process. Figure 10.7 illustrates the use of two different robot types in a construction. The RUBICON simulation can recognize impossible plans, can reveal inefficiencies in task plans, can illustrate the influence of crane placement, and can simulate influences on productivity of human and robot crew interaction. The results of the simulation will assist an engineer or planner to decide on the appropriate mix of robots and human labor crews in the planning stage of a building construction project.

10.1.2 Collision Avoidance

A distinctive feature of robotic construction is the need for a path to be associated with each robot action. The RUBICON program incorporates, within it, an automatic path planner

Figure 10.6 Snapshot of the RUBICON graphical user interface.
which returns a path for each action. A path constitutes a sequence of motion steps selected according to the robot’s motional capabilities and limits, as specified in the rule-based description of the robot’s behavior. A path is designed, using the motion planning algorithm above, to avoid any collision: for each proposed motion step, it is tested whether the robot agent collides with any of the surrounding obstacles, i.e., the building, building components, other robot agents, etc. Only if there is no collision, then the motion step is accepted. For this purpose, the intersection (or product) is determined between the swept volume of the agent, under the particular motion step, and the geometric volumes of the obstacles. In terms of shapes of volume segments, a collision exists between two objects if the corresponding shapes overlap, or one contains the other (i.e., the product is non-empty). Depending on the exact specifications of the volumetric shapes with respect to the

\[\text{Figure 10.7} \quad \text{Dual robot task plan (snapshots from the simulation): (a) a crane positions the wall-panels and (b) a towmotor places the palettized materials.}\]
corresponding objects (that is, whether any safety factor has been considered in the
determination of the geometry or volume of the shapes), shapes may or may not be allowed
to share boundary. Table 5.3 shows that two shapes are neither disjoint nor share boundary,
if at least one of the classes of inner or same-shared segments is non-empty. That is, two
shapes \( a \) and \( b \) overlap, or one contains the other, if \( I_a \neq 0 \land I_b \neq 0 \land M \neq 0 \) (see Chapter
5). If both shapes overlap (or one contains the other), it suffices to find a single segment that
belongs to either \( I_a, I_b \) or \( M \). The algorithm adopted in the implementation of the RUBICON
program uses a slight variation of this result, that ensures the shapes to be disjoint, i.e., they
do not share boundary. Two shapes are disjoint if all boundary segments are classified outer
with respect to the other shape. If an (line segment of) intersection exists between any two
boundary (plane) segments, one from either shape, then at least a part of each boundary
segment must be classified inner or shared with respect to the other shape, and the shapes
necessarily overlap (or one contains the other). If no (line segment of) intersection exists,
then the shapes are disjoint, unless one contains the other. If we make the following
assumptions:

- the initial position of the robot (or any other component) is collision-free, that is,
  the shapes are all initially disjoint;
- the motion steps are small enough, that is, smaller than the minimum dimension
  of any obstacle or building component;
then, we know that one shape can never contain another shape unless they overlapped
before. Thus, an implicit classification is made that is sufficient to ensure that all motion is
collision-free.

10.2 GRAIL

The second application, GRAIL, explores the challenges and the implications that the
algebraic model for shapes poses to computational design. GRAIL currently includes the
following objectives:

2. GRAIL is the brain child of Ramesh Krishnamurti. It is currently being implemented by
M.S. students at Carnegie Mellon University. The work on GRAIL has yet to be documented.
The following exposition only attempts to describe into some extent the aspects of GRAIL that
relate directly to the algebraic model for shapes.
Exploring different ways of interacting with shapes.

Exploring different ways of creating and organizing shapes.

Exploring generative approaches to spatial modeling.

The algebraic model is more than a model for the representation of shapes. It is a concept for spatial modeling that applies as well as a model of interaction for computational design. As such, it has the advantages of being clear, straightforward, yet inclusive: many, if not the most common computational design activities can be expressed using the algebraic model. Here, I leave out any discussion on rules or grammars.

Obviously, creation is a form of addition, i.e. the result of an operation of sum, and deletion is a form of subtraction, i.e. the result of an operation of difference. Less obviously, selection is a combination of addition and subtraction, each on a separate algebra, while deselection is the exact same operation, however, with the algebras switched. Note that creation and selection differ only in the fact that no subtraction is present in the creation process.

Of particular interest is how users can deal with shapes in indeterminate (or emergent) ways: any part of a shape may be considered a shape and be selected. This is quite distinct from the selection process in current CAD approaches where the only objects that can be selected correspond to those (prescribed minimal entities) that have been predefined by the data-structures.

For this, we are developing shape selection methods that derive from cross-algebra operations, where a selector shape may be in a different dimensional algebra than the shape that is ultimately selected. In general, the dimension of the selector shape must be the same or higher than the shape to be selected. If higher, a cross-algebra intersection operator serves to determine the selected shape given the selector. If within the same dimensional algebra, the selection corresponds to the result under the operation of product of the design shape and the selector shape. Otherwise, a cross-algebra intersection operator serves to determine the selection from the selector shape.

Under the algebraic model, a part of a shape can only be distinguished as a separate shape, i.e., selected, if it is subtracted from the original shape. We consider the selection as a shape
in a distinct algebra. Thus, we consider two algebras, for simplicity termed a design algebra and a selection algebra, and the action of selection consists of a subtraction from the design algebra followed by an addition into the selection algebra. Deselection is the exact same process with the source and destination algebras exchanged; contrary to current CAD systems, deselection is not an automatic side-effect of selection.

We have so far considered a design and a selection algebra, but we are not restricted to these two algebras. For example, another algebra may specify a reference shape for assisting in editing or selection: A grid constitutes a reference shape but so may any other shape. Algebras can also be defined by the user in order to layer (and group) elements according to both spatial and non-spatial features. In addition, shapes may have derivational dependencies. A uniform method for interactively and programmatically organizing and manipulating shapes through a tree-like organizational structure is being explored. We eventually hope to handle more general relationships through a graph-like organizational structure.

We are exploring ways of interactively defining spatial rules and rule application. Part of the GRAIL project subsumes spatial grammar interpreters within a 3-D spatial representational environment for modeling, analysis and visualization.

### 10.2.1 Demonstration

Figure 10.8 through Figure 10.18 show the current status of the GRAIL interface. Line and plane segments can be created in a two-dimensional space. Cross-algebra selection tools allow the user to select any part of a shape. The current selection shape can be transformed, under translation, rotation, reflection or scaling, deleted or switched with the design shape. Each of the operations of sum, product and difference can be applied when merging the selection shape with the design. Other modes allow the user to specify the resulting selection shape after the application.
Figure 10.8 Creation tools for line and plane segments. The created elements are currently part of the selection shape.

Figure 10.9 Different drawing modes allow for orthogonal snapping, among others.
Figure 10.10  Operation modes when merging the selection shape with the design, e.g., “difference”. (The selection is discarded.)

Figure 10.11  Different modes specify the status of the selection shape after merging (“sum”) it with the design. The selection is retained (drawn in thin line).
Figure 10.12 “Cutter” selection tools. The cut-out elements are moved into the selection shape.

Figure 10.13 The “switch” tool switches the selection and design shapes.
Figure 10.14 The “bin” tool discards the current selection shape.

Figure 10.15 The transformation tools are translation, reflection, scaling and rotation.
Figure 10.16 The “copy” mode is currently activated: the result of the translation is a copy of the selection shape.

Figure 10.17 Reflection about a line (without copy).
Figure 10.18 Rotation.
The algebraic model and its maximal element representation draw their origins from shape grammars research. However, as a representation scheme, it is potentially more powerful in the general domain of geometric modeling. Not only does this scheme remedy some of the problems that exist with other approaches in solid modeling, e.g., regularity, but it also opens a whole new world of manipulation of figures or shapes in the context of design search and exploration.

Current computational design tools have only limited applicability to design. This is because they restrict more than complement the creativity in the design process. Design search and exploration (Akin, 1986; Smithers and Troxell, 1990; Woodbury, 1991) could provide the necessary support to the designer to facilitate creative thinking in the design process. Viewing design as a generative process of search or exploration is a major paradigm in computational design research, and offers exciting support specifically in the early design stages (Flemming et al., 1993). The algebraic model for shapes particularly supports this view in two ways. First, it is non-restrictive. The model is mathematically uniform for shapes of all kinds and applies to non-geometric elements or attributes as well.
The model provides a natural and intuitive framework for mixed-dimensional shapes. Second, the maximal element representation is essential to the concept of subshapes and emergent shapes. As such, the model directly supports the generation of designs via heuristic rules. On these strengths, a sound theory and implementation of the algebraic model and its representation will enable a new generation of computer-aided design systems.

I conclude with an overview of the main contributions of this thesis and an agenda for future research.

11.1 Contributions

This thesis contains a complete and detailed description of the algebraic model for shapes and the corresponding maximal element representation. The main contributions are:

- A formal and complete definition of the maximal element representation for $n$-dimensional shapes in a $k$-dimensional space ($n \leq k$) and a treatment of the algebraic and geometric properties of shapes.
- The extension of the maximal element representation and shape arithmetic to shapes of volume segments.
- Efficient algorithms for the algebraic operations of sum, product, difference and symmetric difference on two- and three-dimensional rectilinear shapes.
- Exhaustive enumeration of the possible cases for shape recognition on mixed-dimensional shapes.
- The formulation of an agenda of future research necessary to give this representation scheme the power to compete with other geometric models.

11.2 An Agenda for Future Research

In this thesis, I have defined the algebraic model and the maximal element representation and applied them to rectilinear spatial elements. In order to give this representation scheme the power to compete with established geometric modeling schemes, an appropriate representation for non-rectilinear spatial elements needs to be constructed. Even in shape grammars research, the full potential of the model and representation has only been
recognized but not yet fully explored. Limited efforts to incorporate some non-spatial attributes have been noted, such as labels (Stiny, 1980a; Krishnamurti and Earl, 1992), weights (Stiny, 1992) and colors (Knight, 1989). However, no practical method for the inclusion of the non-geometric elements has been set forward that is generally acceptable. Finally, a parametric shape model will allow for parametric shape rules and grammars (Stiny, 1977, 1980a) as well as the specification of classes of similar shapes for shape modeling. A general model for parametric shapes is still non-existent (see also Section 9.2.3).

11.2.1 Surfaces and Curves

While the algebraic model is geometrically complete, i.e., non-rectilinear elements are already covered under the algebraic theory, the representation is lacking in expressive power and is unlikely to gain recognition as a fundamental geometric model without a proper extension to surfaces and curved elements. An appropriate representation for algebraic surfaces and curves needs to be specified that fits within the algebraic model. Most useful surfaces and curves are either algebraic or can be approximated reasonably as algebraic (Hoffmann, 1989a). Examples of algebraic surfaces and curves are polynomial surfaces, Bezier surfaces, and rational B-splines.

The representation of a surface or curve element is two-fold. The carrier of the element is the unbounded surface or space curve that contains the element. The element’s boundary is a lower-dimensional shape on the carrier. Either or both the carrier and the boundary may be curved (non-planar or non-linear). The boundary may be defined by the intersection of any represented surfaces. Since the algebraic operations on shapes are reduced to similar operations on their boundaries, upon splitting the boundary elements into non-intersecting elements, the operation of “intersection” needs to be closed within the set of curved elements (of all dimensionalities).

In developing such a representation, one may be able to rely on past research on the representation of surfaces and curves. Implicit and parametric representations of surfaces and curves are covered in material from both algebraic geometry (Brieskorn and Knörrer, 1986) and geometric modeling (Böhm et al., 1984; Mortenson, 1985; Farin, 1988).
11.2.2 Augmented Shapes

Part of the attractiveness of the algebraic model is its ability to include non-geometric attributes within the model. So far, labeled points as well as weighted and colored shapes have been considered. In all of these examples, the augmented shapes have been derived from shapes of spatial elements by associating symbols, labels or properties, to the elements. Labeled points play an important role in shape grammars. They serve to guide the rule matching process through identification and classification of the rules. They could also be viewed as a semantic extension to what is fundamentally a syntactical expression, i.e., grammars. (Stiny, 1990)

It follows that the algebraic operations, and the underlying part relation, need to be redefined in order to deal correctly with the associated symbols. When considering labels, this can be achieved by an ordinary set approach: the sum of two identical points, each with a set of labels, is the single point with the union of both sets associated to it. It may seem less intuitive for segments of a different dimensionality. Instead, consider shapes augmented with weights or colors. For weights (e.g. line thicknesses), the part relation is obvious and the algebraic operations follow naturally (the sum of two identical segments with different weights is the single segment with the maximum of both weights\textsuperscript{1}). For colors, a ranking may be specified that maps the colors to real values similar to weights or, otherwise, a three-dimensional color coding (e.g., RGB or intensity, saturation and hue) may be considered with an appropriate part relation.

It would be attractive to consider each of these examples, formally, as a Cartesian product of a shape algebra with a non-spatial algebra. However, the algebraic operations do not distribute over the algebras that make up the Cartesian product as is the case in a Cartesian product of spatial algebras only. In the latter case, given two shapes each consisting of a line segment and a plane segment, the sum of both shapes is the Cartesian product of the sum of both line segments with the sum of both plane segments. In the case of colored shapes, the sum of two line segments with different colors that spatially overlap cannot be considered to be the sum of both line segments with a single color that is the sum of both individual

\textsuperscript{1} The weights form a lattice under the relation $\leq$, where the least upper bound is the maximum of two weights and the greatest lower bound is the minimum of two weights.
colors. This only applies to the common segment, any other segment that belongs to only one of both shapes has to retain its original color under a proper algebraic model.

We need to consider a different mathematical formalism for augmented shapes. For this purpose we can introduce a characteristic function to a shape. In constructive solid geometry, a solid can be described as the combination of a set of half-spaces under the Boolean set operations of union, intersection and difference. Each half-space is defined by a characteristic function $g$ with the values 0 and 1, that is, a point $p$ is inside the half-space if $g(p) = 1$ and is outside, otherwise. If the surface bounding the half-space can be expressed as an analytic function $f(p) = 0$, it suffices to define $g(p) = 1$ if $f(p) \geq 0$ and $g(p) = 0$ otherwise.

Similarly we define a characteristic mapping $f_a$ for a shape $a$. However, since a shape is not considered a point set, unless its spatial elements all have dimensionality 0, its definition is slightly different. Consider the set $U$ of all finite arrangements of $n$-dimensional hyperplane segments of limited but non-zero measure in a $k$-dimensional space, for a given $n \leq k$. The function $f_a$ is defined from a subset of $U$ to the set $\{0, 1\}$ such that for any element $s$ of $U$:

$$f_a(s) = 1 \text{ if } s \text{ is a part of } a,$$

$$f_a(s) = 0 \text{ if } s \text{ and } a \text{ are disjoint (their product equals 0).}$$

Note that $f_a$ is not a function from $U$ as it is not defined for all shapes of $U$; for a general shape $s$, it only holds that $f_a(s \cdot a) = 1$ and $f_a(s - a) = 0$, with $(s \cdot a) + (s - a) = s$.

In order to allow for a characteristic function, consider $\Delta$ the set of infinitesimally small elements of $U$. (Actually, it suffices to choose the elements of $\Delta$ small enough such that $f_a(s)$

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure111.png}
\caption{The operations of sum and product over weighted shapes of line segments.}
\end{figure}
exists, i.e., has a unique value, for every element \( s \) of \( \Delta \). Then, \( U \) can be considered a power set over \( \Delta \), with any two elements in \( \Delta \) disjoint. Define \( g_a \), a mapping from \( \Delta \) to \{0, 1\} such that \( g_a(s) = f_a(s) \). Given the choice of elements in \( \Delta \), \( g_a \) constitutes a function. Let \( g_a \) denote the characteristic function for \( a \) with respect to \( \Delta \). Consider the set \( G \) of all characteristic functions \( g_u \) with \( u \) an element of \( U \) and define the function \( f \) from \( U \) to \( G \) with \( f(a) = g_a \) that maps each shape onto its characteristic function for \( \Delta \).

A range \{0,1\} distinguishes conceptually between shapes that are a part of (1) and shapes that are disjoint (0). When considering weights, any part is assigned a weight and we must distinguish shapes with different weights. If we consider a weight to be represented as a positive real value, the range of weights constitutes \( \mathbb{R}^+ \), the set of positive real numbers. Any infinitesimally small element \( s \) of \( \Delta \) is assigned a single weight and we can consider the characteristic function \( g_a \) to be a (step-wise constant or otherwise) function from \( \Delta \) to \( \mathbb{R}^+ \).

Upon embedding weighted shapes in the Euclidean space, the result is similar to a spectral (discontinuous) function that specifies a “height” for every point. The characteristic function of two weighted shapes equals the “sum” of the characteristic functions of both shapes, where the operation of sum is defined for the algebra of weights, that is, the sum of two weights is the maximum value of both weights.

In general, when augmenting the shapes with non-spatial information, we only need to redefine the range for the characteristic function, e.g., we create one or more extra dimensions (e.g., three in the case of colors) that define the range space for the characteristic function. In the case of labeled shapes, for a given set of labels \( L \), the range of the characteristic functions is the power set of \( L \) and the algebraic operations correspond to the set operations.
## Glossary of Mathematical Notation

<table>
<thead>
<tr>
<th>Logic</th>
<th>Shapes</th>
</tr>
</thead>
<tbody>
<tr>
<td>∀</td>
<td>0 empty shape</td>
</tr>
<tr>
<td>∃</td>
<td>+ sum</td>
</tr>
<tr>
<td>∃!</td>
<td>- product</td>
</tr>
<tr>
<td>:</td>
<td>− difference</td>
</tr>
<tr>
<td>¬</td>
<td>⊕ symmetric difference</td>
</tr>
<tr>
<td>∧</td>
<td>≤ part</td>
</tr>
<tr>
<td>∨</td>
<td>= equality</td>
</tr>
<tr>
<td>⇒</td>
<td>$B[]$ boundary</td>
</tr>
<tr>
<td>⇔</td>
<td>$\Gamma()$ constructor</td>
</tr>
<tr>
<td></td>
<td>$\Delta()$ neighborhood</td>
</tr>
</tbody>
</table>
### Glossary of Mathematical Notation

<table>
<thead>
<tr>
<th><strong>Sets</strong></th>
<th><strong>Vectors</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>( \times )</td>
<td>( \vec{p} )</td>
</tr>
<tr>
<td>( \mathcal{P}())</td>
<td>cross product</td>
</tr>
<tr>
<td>( \emptyset )</td>
<td>( \cdot )</td>
</tr>
<tr>
<td>( \subseteq )</td>
<td>element</td>
</tr>
<tr>
<td>( \subseteq )</td>
<td>identicalness</td>
</tr>
<tr>
<td>( \cap )</td>
<td>subset</td>
</tr>
<tr>
<td>( \subseteq )</td>
<td>proper subset</td>
</tr>
<tr>
<td>( \cap )</td>
<td>intersection</td>
</tr>
<tr>
<td>( \cup )</td>
<td>union</td>
</tr>
<tr>
<td>( \setminus )</td>
<td>difference</td>
</tr>
</tbody>
</table>


