

ASPECTS OF THE GEOMETRIC REPRESENTATION OF KNOWLEDGE FOR COMPUTER AIDED DESIGN

by

Steven Frank Bridge
B.Sc.(Special Eng. Programme)

1988

A thesis submitted for the degree of Doctor of Philosophy of the University of London
and for the Diploma of Membership of the Imperial College of Science and Technology

Department of Computing
Imperial College
London, SW7

ABSTRACT

If Artificial Intelligence is ever to enter the field of assisting Computer Aided Design, it must first overcome a shortcoming. This is the need to represent a certain type of information frequently required in mechanical engineering design, namely any information which is best represented by means of geometric entities in a real number space. This type of information includes the description of the geometry of engineering components as well as the graphs of functions describing aspects of the performance of the design which may require more than three dimensions.

An outline of a system capable of handling this type of information is described. This system is a generalisation of solid modelling and consists of extending the model space to n -Dimensions ($n > 0$), and allowing infinite objects. Centrally, a "Join" operator is defined which maps a set of points into their convex hull. This is augmented by a hierarchical method of representation based on "excursion" boxes which provide a variable level of detail for accessing the geometric models.

The need for flexibility arises from the observation that a collection of geometric entities (a model) may be interpreted as representing the (whole or partial) shape of a physical object or as the graph of a function relating several engineering variables. In the first case the usual solid modelling operations (for example, intersection detection) are required, whereas in the latter case such operations as function application or composition would appear more relevant. The proposed system is capable of both kinds of operations and therefore extends geometric modelling into the geometric representation of knowledge which is illustrated in the Thesis.

Some of the more complex algorithms were implemented and the surrounding implementation issues discussed. Finally a few examples illustrate the important components of the proposed system.

ACKNOWLEDGEMENTS

I would like to express my thanks to Professors I.Aleksander and A.Medland for their support, their guidance and the numerous helpful discussions throughout the course of this research.

Financial support for this research was provided through the generosity of Dowty Rotol Ltd, Gloucester, England under their "Computer Aided Design Research Assistantship" scheme, and I would like to take this opportunity for thanking both Mr.J.Bridge and Dr.D.Wood for their invaluable assistance with administrative matters.

Finally I would like to express my thanks to my parents for their encouragement and moral support.

TABLE OF CONTENTS

ABSTRACT	2
ACKNOWLEDGEMENTS	3
LIST OF ILLUSTRATIONS	6
Chapter 1 INTRODUCTION	7
1.1 Definition of Design	8
1.2 Overview	8
Chapter 2 REPRESENTATION OF GEOMETRIC AND DESIGN	
INFORMATION	10
2.1 Design Methodology	10
2.2 Computer Aided Design	19
2.3 Advanced Systems for Design	23
2.4 Qualitative Reasoning	39
2.5 Design Information Summary	46
Chapter 3 THE GEOMETRIC REPRESENTATION OF	
KNOWLEDGE	51
3.1 Objectives and Overview	51
3.2 Generalised Polyhedral Modeller	56
3.3 Excursion Boxes	63
3.4 Constructive Definitions	69
3.5 Plurality of Interpretation	74
3.6 Qualitative Representations	78
3.7 Geometric Knowledge Summary	83
Chapter 4 IMPLEMENTATION ISSUES	88
4.1 Overview	88

Table of Contents	5
4.2 Data Structures	89
4.3 Algorithms	101
4.4 Implementation Summary	127
Chapter 5 GEOMETRIC KNOWLEDGE EXAMPLES	130
5.1 The Addition Plane	131
5.2 The Multiplication Surface	136
5.3 The Excursion Box Hierarchy	142
5.4 Possible Geometric Reasoning	147
5.5 Examples Summary	151
Chapter 6 CONCLUSIONS	153
6.1 Concluding Summary	153
6.2 Future Work	155
6.3 Consequences	155
REFERENCES	161
APPENDIX A Convex Set Theory and Topology	169
APPENDIX B Comparison of Convex Sets	178

LIST OF ILLUSTRATIONS

Figure 5.1	The addition plane.	130
Figure 5.2	Applying the addition function.	132
Figure 5.3	Applying the addition mapping.	134
Figure 5.4	The multiplication surface.	136
Figure 5.5	Bounding tetrahedron T_1	137
Figure 5.6	Constructive definition—Part 1.	138
Figure 5.7	Constructive definition—Part 2.	139
Figure 5.8	Constructive definition—Part 3.	140
Figure 5.9	Two intersecting excursion boxes.	142
Figure 5.10	Refining the excursion boxes.	143
Figure 5.11	Refined excursion box overlap.	144
Figure 5.12	Two objects in contact.	144
Figure 5.13	Two disjoint objects.	145
Figure 5.14	Two intersecting objects.	146
Figure 5.15	Initial geometric model G	147
Figure 5.16	Geometric model of the slope of G	148
Figure 5.17	Detailed model of the slope of G	148
Figure 5.18	After the comparison with B_R	149
Figure 5.19	A new partition of G	149

Chapter 1. INTRODUCTION

Design as an age old human activity is also a prime example of advanced intellectual activity and thus an ideal subject of investigation for the advancement of Artificial Intelligence (AI) while simultaneously being of practical interest.

The reason for which design is such an interesting area of study is that it consists of the transition between ill definable 'real world' information (for example, an outline of a need or an imprecise specification) to a much more precise form of description (of a product) which comes closer to the formal representation methods used within AI. By considering the 'output' from the design process and attempting to infer the kind of reasoning activities that may have lead to them, AI could use the area of design as a stepping stone towards expanding its area of applicability further in the direction of the mainstream of human concerns.

The practical interest stems from the need for better 'design aids'. This, in turn, can be traced to the combination of two factors, namely the increasing complexity of modern engineering artifacts in conjunction with the increasing competitiveness of a world market place whose demands reduce the amount of time available for the development of these more sophisticated products.

Before proceeding with the overview of the thesis a brief clarification of the meaning of the term "design" is required.

1.1 DEFINITION OF DESIGN

The background against which this discussion is situated is the design of mechanical engineering systems, with the main emphasis being placed on the representation and use of geometric (and related) information. In this context “design” can be defined as the purposeful activity of deriving a specification of a product from the description of a need it is supposed to meet [LEWI81,POPP86]. In other words a *description* consisting of a set of requirements has to be transformed into a sufficiently detailed *description* of an artifact enabling it to be manufactured [JAKO85a, McDE78] (The quest for the *true* description of the nature of the design process will be considered in greater detail in Chapter 2).

1.2 OVERVIEW

As suggested above, this thesis investigates aspects of the use of geometry to represent certain kinds of ‘information’ or ‘knowledge’, as found in engineering design offices, and considers some of the resulting implications for incorporating Artificial Intelligence (AI) techniques into future Computer Aided Design (CAD) systems.

As the application of AI techniques to engineering problems is a development which has only emerged relatively recently, the practical aspects of any particular design process remain beyond the reach of this investigation which therefore conducts its study at a more appropriate and fundamental level.

The thesis is roughly divided into two major Sections, the theoretical considerations (Chapters 2 and 3), and more practical implementation problems (Chapters 4 and 5).

Chapter 2 examines the state of the art in those areas of research which are relevant to the area of design as described above. Section 2.1 investigates the broad area of research into the nature of design processes and design methodologies, Section 2.2 explores computer aided design as a means of representing geometric information and Section 2.3 considers some of the advanced “knowledge based”

systems used for representing and applying design knowledge. Next Section 2.4 considers qualitative reasoning/simulation systems. Finally, Section 2.5 discusses the different kinds of knowledge that were found and presents the geometric approach to the representation of certain kinds of engineering information.

Chapter 3 essentially consolidates and expands the ideas presented in the last Section of Chapter 2. In particular Section 3.1 provides a statement of the objectives of a system for the geometric representation of information together with an outline of the proposed approach. The next three Sections (3.2 to 3.4) then present the individual components of the proposed system, and Section 3.5 describes the flexibility of interpretation which elevates geometric models to the rank of geometric knowledge. Section 3.6 briefly considers the integration a qualitative representation scheme before Section 3.7 provides a summary of the ideas presented in that Chapter.

Chapter 4 is mainly concerned with implementation issues. Section 4.1 provides a brief overview of the Chapter before Section 4.2 proceeds to investigate possible choices of data structures and their implications. Section 4.3 then provides a detailed description of the major geometric modelling algorithms before Section 4.4 concludes with a statement of the extent to which these were actually implemented.

Chapter 5 contains a few examples that illustrate the features of the proposed modelling system. These start with a simple model of geometric knowledge in Section 5.1 and proceeds to a more complex constructively defined one in Section 5.2. Section 5.3 illustrates the hierarchical aspect of the proposed system and Section 5.4 briefly considers the possibility of geometric reasoning before Section 5.5 provides a summary.

Finally, Chapter 6 concludes by providing an overall summary in Section 6.1, outlining directions of future research in Section 6.2 and considering possible consequences in Section 6.3.

Chapter 2. REPRESENTATION OF GEOMETRIC AND DESIGN INFORMATION

The activity of design lies at the intersection of many disciplines. This Chapter tries to reflect the current state of knowledge about the process of design itself before considering those areas of current research that are most relevant to it. However, due to the interdisciplinary nature of design and the amount of interest in it and related subjects, the relevant literature is both large and scattered. Therefore the following Sections can only briefly explore each area and highlight those aspects which appear to be most relevant in the context of this thesis.

2.1 DESIGN THEORY AND METHODOLOGY

The design process (though it may be more accurate to refer to “design processes”) consists of a large number of problems to solve and decisions to make. When satisfactory solutions have been found, certain patterns can be identified within the activities which resulted in those solutions. Design theory and methodology are respectively the propositions that these patterns can be listed in the form of models of the design process or as sets of rules, or methodologies, to aid future design work [FINK83]. The motivation for these studies seems to arise from the need for some general principles for the teaching of design, the organisation of larger design projects and the provision of aids to designers, in particular with respect to the use of information technology. This last point is the one which caused the investigation of Design Theory and Methodology in the context of this research.

It follows that the concepts presented in this Section are centred on the two themes of *models of the design process* and *methodical design*. The difference between them is that the former are created from the perspective of studying,

or trying to understand, “design”, whereas the latter are more concerned with guiding and helping designers. Both will therefore reveal slightly different aspects of the nature of design processes. However, it is first necessary to investigate the process of decision making as this is central to any design activity.

2.1.1 Decision Making

Many authors [FINK83, SAMB83, MOST85, MEDL86, ULLM87] have investigated this activity, and arrived at remarkably similar descriptions. The emerging representation of decision making seems to be as a sequence of individual stages whose labels vary according to the author.

The purpose of each decision process is to either satisfactorily deal with the decision goal or prove that it is not practical [BATA83].

The description and number of stages in decision making ranges from a brief synthesis—analysis—evaluation cycle to the more detailed description of the kind shown below :

1. Information gathering (using the requirements specification provided by the preceding design stages) and thus framing the issue or goal to be decided.
2. Establishing the criteria to be used for comparing the alternatives.
3. Generation or synthesis of alternative design solutions.
4. Analysis of these design solutions to determine their properties.
5. Evaluation according to the selected criteria.
6. A final decision step in which either a design solution is kept, or all are rejected and some of the earlier steps, of this particular decision process or of some preceding one, repeated. The latter case effectively involves retracting some earlier decisions.

The clarity of the above structure in any design decision process is affected to a large extent by the scale of the corresponding design project. A large scale project involving many engineers and different organisations will tend to exhibit more

formal structuring and decision making as well as explicit division into stages, whereas with single designer working on a small task the different decision steps will merge indistinguishably into each other.

It is further possible to remark [MOST85] that the decision making process is recursive in the sense that each of its stages is a small decision making process in its own right and composed of the sequence of steps shown above. This can be illustrated by considering that, for example, the steps of synthesising alternatives or performing an analysis themselves involve making a decision as to when to stop, that is deciding whether the alternatives or information generated so far will be sufficient.

Having briefly described the decision making activity, the next Section will explore its evolution during the design process itself.

2.1.2 The Design Process, its Problems and their Representation

Observations of engineering design work [ARCH85] suggest that the designer's focus of attention continually loops around three centres, namely, the project, the product and the current problem.

When concentrating on the project the designer mainly seems to be planning his next activities. This may involve measuring the current progress with respect to some adopted design methodology.

When the designer's attention is focussed on the product he will essentially be trying to reconcile, while simultaneously developing, two distinct views of his design. These are his increasing insight into the existing constraints and his gradually more concrete proposal of a design that will meet those constraints.

The final focus of attentions is on the design problems which may occur at both the product and project level. A problem occurs when there is an unsatisfactory current situation, a desired goal situation, and some obstruction which currently prevents the goal from being reached.

The obstruction to progress is in general due to a lack of some kind of information. For example [ARCH85, PAHL85], the means of reaching the goal are unknown (synthesis problem), or, though known, they are so numerous that a systematic evaluation of all combinations is not feasible (selection and combination problem, for example when selecting components from databases of standard parts [CULL86, VOGW86]), or the goals may be vague and unclear as in the case of using a known, but not yet applied technology (search and application problem).

Further important properties of problems [ARCH85, PAHL85] are the *complexity* with its resulting lack of overview and control arising from a large number of strongly interrelated components, the *uncertainty* due to an ill defined starting position (for example, obscurity about the nature of the relations contained in some function or mechanism) which means that some design goals cannot yet be finalised, and the *influence* of a solution component on the whole or part of the design which cannot be evaluated immediately (for example, the impracticality of a proposed approach due to overlapping constraints may not be immediately evident) and is often only fully understood once the design has been completed.

The influence that any one decision has on the rest of the design process is difficult to assess because of the nature of the interdependency of the design decision structure. Each design decision is influenced by both the current issue (or goal) being decided and states of affairs which have come about, at least in part, as the result of earlier decisions. The state of affairs brought about after a decision is made will alter the design in some ways so as to satisfy the goals mentioned in the decision, but will also refine existing goals as well as introducing new goals for more detailed elements of the design and thus require further decisions to be made [BATA83].

Furthermore, each design decision is influenced, not only by the results of more preliminary decisions, but also by new knowledge being obtained from decisions on a similar or even more subordinate and detailed level. Thus the results of decisions at lower functional levels may influence already worked out zones at a higher level, and introduce many corrective and repeated sequences of steps into

the design process [PAHL84, ANDR85, ULLM87]. This is a consequence of the final step of the decision making process as described above.

This means that not only will every decision depend on, and itself influence, other decisions, but also that the interdependence of the design decisions for a given design project cannot easily be determined in advance, and that, in terms of the finished product, many decisions may appear arbitrary or at least are not obvious because they are the consequence of the sequence of decision making. It follows that the "object level" understanding of a design, of how and why it works, is separate from the understanding of the design process which created it.

To obtain a better understanding of a design the interdependency of decisions, as well as the existence of any external factors that may have influenced a design, should be made explicit [BATA83, JAKO85a, MOST85, BRAI86]. Changes in these factors may change the validity of some of the decisions made in the design process and thus be important for maintenance, further development or the "replay" of a design. This also applies to any design alternatives that were investigated and the reasons for which they were rejected.

A tree structure has been suggested for the representation of design goal interactions [BATA83, MOST85]. This would allow the consequences of a given decision to be followed, and conversely those results to be traced upon which a particular decision depended. Nevertheless, there are several problems with this approach.

The first of these is associated with the nature of goal interactions [MOST85]. A tree structure can represent subgoal-supergoal relationships that involve conjunctive, disjunctive and sequential relationships. However there are also more complex kinds of goal interactions such as independence (when different goals do not affect each other), cooperation (where achieving one design goal assists in achieving others), competition (when one goal can only be achieved to the detriment of others and thus often leading to a compromise solution), and interference (which is a less severe form of competition in which all goals can be

achieved provided that the other goals are taken into account while working on any particular one).

A further problem with goal tree like representations is that many decision cannot be represented explicitly, or may depend on subjective elements of which a designer is not explicitly aware [BATA83]. Also maintaining such a description of the design process is not a generative aid to design, but more akin to a system of documentation.

During the design process, the issues being decided tend to set out from a general and abstract overview of the whole design project and move progressively towards increasingly more localised, concrete and detailed development of individual components [MACC85, ULLM87]. Indeed the design process is conventionally divided into a linear sequence of the four design phases of *specification development*, *conceptualisation of feasibility*, *layout design*, and *detail design* [ULLM87]. However this does not mean that the design activities should be represented as a linear sequence of activities [EAST78] as this would impose an order on design decision making and therefore also on what decisions could constrain others (see above discussion).

It has been suggested [MACC85] that it is the complexity of design tasks that forces use of abstractions in order to make individual design stages more tractable as a result of reducing both the number of independent parameters and their interactions. Another contributing factor is certain to be the almost complete absence of detailed "object level" information at the outset of a design project.

However, advice on how a design process should proceed is available in the form of Design Methodologies which are investigated in the next Section.

2.1.3 Methodical Design

The aim of design methodologies is to provide a strategy for the engineering design process. Although many of the design methodologies are dominated by the area of expertise or specialisation of their authors, they do share a common structure which has been captured in some broader unifying text books such as the one by G.Pahl and W.Beitz [PAHL84].

Design Methodology follows the four stages (specification development, conceptualisation of feasibility, layout design, and detail design) of the design process while providing advice or suggesting representation schemes appropriate for each.

In particular, the first design stage of clarification of the actual design task can be supported by various techniques such as the use of check lists, lateral or divergent idea generation, or the use of formal specification or description languages [FINK83].

The suggested approach to the conceptualisation of feasibility stage of the design process can be characterised as being very systematic. It consists of first abstracting the requirements of the specification for the purpose of establishing the overall function of the system, as well as clarifying the system boundary together with all relevant inputs and outputs.

In the next stage the design is expanded into alternative function structures [PAHL84]. This consists of a process of decomposition and abstraction of the design task where the engineering system is viewed as a set of entities without (at this stage) considering the form of possible realisations [FINK83].

As there is no unique logical method of decomposition and this process may lead to very different solutions. However, there appear to be sets of primitive functions out of which the total system function can be built up [FINK83]. For example, in the initial stages the system can be considered as a set of black boxes linked by flows of energy, materials and signals [PAHL84]. Alternatively, the functional aspects of a technical system may be represented by process orientated

formalism reminiscent of petri nets [JOHN85], which takes into account physical compatibility constraints.

The generation of concepts that this requires can be stimulated through the exploration of existing design concepts, or the use of systematic modification (of the proposed realisation of an individual function, or of the function structure itself), negation (for example through removal or reversal of a component) or the analysis of characteristics (with the emphasis being placed on the enhancement of desirable features and the removal of limitations) [FINK83, ROTH85]. For example, the functional description may first be critically assessed in terms of what the function is, why and how it is performed and what its fundamental limitations are.

Once the function structure has been established, the emphasis shifts towards finding alternative physical phenomena which can be used as the basis for implementing each individual function. When this is completed, the layout phase of the design along with the detail design of the shape and material of individual components can be started.

During the layout phase of the design process the essentially form and dimension independent solution concept is converted into the final layout where the required key surfaces and motions have been determined, all mechanical components given at least a preliminary shape and an outline of the manufacturing and assembly processes established [ANDR85].

Finally the process of evaluation and decision making has, provided that the criteria are fixed, at least an outline of a rational methodology in the form of operations research or management science [FINK83].

2.1.4 Summary

The models of the design process and the ideas behind the methodical approach to design complement each other to some extent in the attempt to describe the kind of information processing activities which constitute a design process.

In particular, the design methodologies suggest, as far as is possible, the type of information and corresponding representation format or primitives to be used at each “stage” of the design process, while simultaneously ensuring a step by step progression from the initial qualitative design overview to the final quantitative detailed product [PAHL84].

However, the immense complexity of design tasks is also clearly illustrated. When taken as a whole, these still appear to be both poorly understood (in particular with respect to additional issues, which have not been considered here, such as the coordination of many individuals in a design team [SAMB83]) and beyond the current level of CAD technology [ULLM87]. The main criticism seems to be that methodical design is about what the designer *should* do, and a guide to *what* is done, but gives no information on *how* [ULLM87].

The following Sections will investigate the use of computers in the design process, starting with area of Computer Aided Design (CAD).

2.2 COMPUTER AIDED DESIGN

The ultimate objective of Computer Aided Design (CAD) systems may be described as achieving an intimate interchange of information with the designer in terms of both representing and acting upon any thought structures the designer may wish to express [MACC85].

However, current CAD systems might best be described as “the use of interactive graphics to help solve design problems” [ULLM87]. For mechanical engineering the major application areas are as a sophisticated drafting tool with additional benefits being derived from smoother communication and more consistent organisation of data, and as pre- and post- processors for analytical software packages and the visualisation of complex shapes and data.

Current CAD systems are mainly concerned with the processes of generating, evaluating and modifying a model of an object through the interaction with a computer and thus primarily a system of documentation of the finished design as opposed to a help with the actual design process [OSHU83, SAMB83].

Computer Aided Design systems usually consist of drawing or solid modelling software more or less loosely linked to auxiliary packages for stress or kinematic analysis or tool path planning.

Solid modelling, as an integral part of many of the more sophisticated Computer Aided Design systems, is responsible for maintaining informationally complete descriptions of the geometry of an engineering component. This overcomes the deficiencies of earlier “wire frame” CAD systems in which the models were ambiguous in the sense that is different solid objects could be represented by a same wire frame model.

Although there are many different solid modelling schemes [REQU83, CASA85] the two principal ones are *constructive solid geometry* (CSG) and *boundary representation* (B-rep).

Constructive Solid Geometry is based on the idea of building complex solids out of simpler ones through the ordered use of modified set theoretic operations such as union, difference and intersection. The starting position is a set of instances of built-in primitive solids such as blocks, spheres and cylinders, which form the leaves of a binary tree. The interior nodes of the tree each contain one of the set theoretic operations and represent intermediate solids obtained by applying that set operation to the two intermediate (or primitive) solids just below it in the tree. The root node of the CSG tree then corresponds to the complete solid. Performing an operation (such as displaying or computing a mass property) on the model involves applying an algorithm to each of the primitives and then combining the results for each intermediate solid until the entire model has been processed (that is, the root of the CSG tree has been reached).

The B-rep representation scheme is very different. A solid is represented in terms of its boundary or enclosing surface. The boundary itself consists of the union of faces, with each face in turn represented as the union of its bounding edges together with data defining the surface in which the face lies. The construction operations in B-rep systems must ensure that the boundary is well formed in the sense of fully defining a solid object.

Two further solid modelling schemes of interest are *Analytic Solid Modelling* and *Octrees*. Analytic solid modelling [CASA85] represents a solid as a non-overlapping collection of solid components known as hyperpatches. These are the extension to three dimension of the two dimensional parametric surface patches used in some B-rep systems. The hyperpatch represents a solid by a tri-cubic polynomial mapping from a parameter space, the *solid* unit cube, into three dimensional model space. The result is a solid which is described on its interior as well as on its bounding faces which, in general, are sculptured surfaces. The ability to describe the interior of the solid as well as its external shape makes this form of solid modelling particularly suited to complex analysis (or visualisation) problems involving properties which are distributed anisotropically across the interior of the solid.

The Octree representation of solid objects [GUER86] is again very different from any of the above schemes in that the whole three dimensional space is represented. An Octree is an eight-ary hierarchical subdivision tree of a finite, cubic volume element. The cube at the root of the tree represents the entire universe. This is decomposed recursively into eight smaller cubes of identical size, one for each octant of the parent cube. Each of the smaller cubes is then decomposed in a similar fashion until some specified resolution has been reached. Each cube in the Octree represents a region of space which can either be partially occupied by (part of) the solid object, completely occupied or not contain any part of the solid. Cubes which are either entirely occupied or entirely empty need not be subdivided any further and thus form the leaves of the Octree. Any cube that is partially occupied will be further subdivided, if necessarily, up to the resolution of the Octree. The advantage of the Octree resides in its very uniform and simple representation of the region of space occupied by an object. This greatly reduces the complexity of many of the modelling algorithms, but only at the expense of storing very large amounts of data which will, in general, be required to represent a non-trivial object in sufficient detail.

The input of data into solid modelling systems is mainly done interactively through the use of graphic interfaces. The actions available to the user are typically to create a new object, alter it and obtain some of its section or volume properties. However, the range of operations on solids supported by most modellers is often very limited and typically only contains transformation by rigid motions and joining by boolean operations. The availability of additional operations depends on the type of representation used in the solid modeller. For example B-rep systems may offer a sweep operation which moves (sweeps) a two dimensional shape along the third dimension to define a solid object. Solid modelling CAD systems are powerful and sufficiently diverse to meet the basic geometric representational needs encountered in engineering design. There is, however, no scheme for representing solids which is uniformly better than all the others, and the process of defining engineering components is still cumbersome with most systems and therefore still a subject of further research [CASA87, ROSS86]. Most systems also provide programming language type of facilities

which are mainly used for the purpose of defining parameterised object families where typically the shape of the object is determined but its dimensions are not (yet) fixed.

The very precision in the manipulation of three dimensional geometry offered by these solid modelling CAD systems makes them unsuited for the initial design stages as the operator really needs to know in advance and in some detail what he wants to do before approaching the system. It is for this reason that the design process support offered by these systems lies mainly in the final detailed stages of the design process.

More sophisticated computer based systems for design which are still the subject of current research are considered in the following Section.

2.3 ADVANCED SYSTEMS FOR DESIGN

The desire to augment the usefulness of basic Computer Aided Design systems has led to a spectrum of research starting with the small scale abstraction of geometric features, through more substantial additions to CAD systems and culminating in full scale knowledge based expert systems for more or less autonomous design. The following Sections will investigate each of these three approaches and also explore some additional systems which do not fit easily into this spectrum.

2.3.1 Design with Geometric Features

The representation of the geometry of the shape of an engineering component inside a CAD system is not ideal for the purpose of design decision making. An interesting approach centres on the idea of *geometric features* of a design, which are aimed at representing more abstract, higher level knowledge about the design than the details of geometric primitives used to describe objects in the typical CAD system. For example, the coordinate information of points, curves, and surfaces would form the lowest level of representation. At a higher level the same information could be presented as several slabs joined by blending surfaces, while at a still higher level the object might be identified as a box, whose function was to enclose some piece of equipment [DIXO87, DIXO85]. In particular, the low level information can be obtained easily from the higher level description in terms of feature parameters. This means that the user can rapidly effect complex alterations of a CAD model through changes made in terms of its features [BRAI86] and thus enjoy an improved (higher level) man-machine interface. A feature may therefore be defined as a geometric form or entity which is relevant to the design task and whose presence as a primitive facilitates the design process [DIXO87]. Thus to be effective, a sufficient library of relevant features must be provided.

The concept of feature is clearly related to the idea of parametric representation of components as already available on some CAD systems. The difference is that features only represent certain parts or aspects of a component, and that a

whole component would in general be made up out of many specific instances of different kinds of features.

The concept of feature remains fairly fluid as the level of abstraction provided may range over a fairly wide spectrum, from, for example, a description in terms of “walls” and their intersections, through primitive shapes (reminiscent of the primitives provided in solid modelling systems, except that they are chosen to be more relevant to the particular design problem) which can be combined to yield the desired component, up to a classification of parts into broad families of similar geometric shapes such as “box”, “U-channel”, or “angle bracket” [DIXO87].

By restricting a design component to consist exclusively of an amalgamation of the provided features, the theoretical issues arising out of data structures which do not correspond to any physically realisable objects are avoided because they simply cannot be expressed. Whether the associated restriction on the variety of useful designs that can be expressed is always intolerable remains to be seen [DIXO87].

With the multitude of factors that have to be considered during design, it is also likely that what constitutes a *relevant* feature will depend on the circumstances and thus change during the design process. For example, for the purpose of obtaining a weight estimate of some component some small geometric features (for example, a small lubrication channel) may be ignored, but may become of vital importance for a strength calculation as it may give rise to a stress concentration. This also leads to the possibility that some low-level geometric entities may belong to different features, i.e. features may overlap, a phenomenon which will increase the complexity of propagating design changes. Consider the case of a box which could be described in terms of a base plate and four walls or as a cavity subtracted from a solid block - changing the cavity on its own will affect the wall thickness as a side effect which may or may not be desirable. Furthermore, there is the possibility that for a practical design system an intractable number of features might, in fact, be required. Thus the area of geometric features is one which still contains many open questions.

The link with design theory is the desire to develop abstractions which can reflect the designer's concerns which focus on parts of a component as opposed to treating the component as a whole. The advantages to be gained from this approach are twofold; firstly to improve the communication between the designer and the modelling system and secondly as a (future) attachment point for additional design information such as the reasons for which a component was given its particular shape [BRAI86, JAKO85a].

The idea of augmenting a CAD system through the provision of geometric features can be taken a step further as described in the following Section.

2.3.2 Enhanced Computer Aided Design Systems

Enhanced computer aided design systems have been constructed based on the observation that many components or mechanisms consists of clearly identifiable design elements, each with its own design rules and optimisation procedures. The idea of *design element* is more advanced than that of geometric features in the sense that each design element typically consists of several components and is combined with a program for both computing its size and optimising it with respect to some selectable criteria.

Design is then the process of combining such design elements which are selected according to compatibility (for example, size and materials) and optimisation (for example size, weight, cost) criteria and automatically dimensioned to suit the particular design requirements. The enhanced CAD system is given a modular structure along lines similar to that of a catalogue of standard parts, with as many different (sub) programs as there are types of design elements. Each of these programs is responsible for the whole of the detail design and optimisation of its type of element, including, in some cases, the production of a detail drawing.

There are two working systems based on this idea : first the SICAM system [GUIL85] whose available design elements currently include a variety of bearing, sealed joints, gear wheels, splines, bolted connections, and the like, and secondly

the JIGS system [MILL85] which is specialised on the design of jigs and fixtures and thus includes a variety of clamping and positioning elements.

Both systems are interactive, and in the case of the JIGS system, requires user interaction mainly because "the computer does not know it has a component of a certain form; it only knows it has a series of lines, arcs, circles, etc with certain spatial relationships" [MILL85]. That is, although the computer has a full description of the geometry of the part for which the jig is needed, it is not capable of obtaining an "overall shape" which the human designer would use to guide the positioning of the key elements of the jig.

CAD systems have also been expanded in a different direction as for example in the HMADES-1 system [KITA84]. This system comprises a hierarchical (and therefore treelike) modeller of numerical relationships occurring between individual numerical variables which in fact also allows loops in its structure (that is the treelike hierarchy turns into a graph). The hierarchy is used to describe relationships between the individual components of some mechanism or system. Propagating changes of a numerical parameter in a hierarchy is straightforward. However, when a structural loop occurs, a special algorithm has been provided to allow updating of the parameters involved in the loop once the dependent variables have been identified.

This algorithm can be used for both dimension matching (i.e. maintaining a given relationship between some dimensions when one of them is changed) and motion analysis (where the updated quantities are not interpreted as dimensions but correspond to the positioning variables of individual elements in a mechanism). The noteworthy issue here is the dual application of the algorithm which suggests that reasoning about geometric quantities is but a particular application of reasoning about real valued numerical quantities.

The more advanced form of computer support for the design process takes the form of Knowledge Based Expert Systems which are considered in the following Section.

2.3.3 Knowledge Based Expert Systems for Design

Knowledge Based Expert Systems are a practical application of Artificial Intelligence research. The field of Artificial Intelligence is concerned with the construction of more or less successful computational models of intelligent human activities such as perception (speech recognition, image analysis) and reasoning (language understanding, theorem proving, game playing and the like). Knowledge Based Expert systems are additionally concerned with the process of making decisions and often can provide near natural language explanations of the course of their deductions when the user so desires.

There are three main constituents of a typical rule based expert system. The first is its knowledge base. This consists of the expertise of one or more human experts represented in a formal manner often as a collection of rules in the form "if condition then conclusions". The second element is the data base describing the current state of the problem under consideration. The final element is the "inference engine" which will systematically apply the relevant rules in the knowledge base to the current problem description in order to respond to the user's commands.

The application of Knowledge Based Expert Systems to mechanical engineering design is a fairly recent development [DYM.85], as the many of the previous areas of applications involved principally diagnostic tasks. The next Sections describe some of these systems.

2.3.3.1 The COPART and CARTER systems

The CARTER [REYN84] and COPART [AQUE83a, AQUE83b] systems are the oldest examples of Knowledge Based Expert Systems applied to mechanical engineering design. Both are essentially written in FORTRAN, the rule base is in near natural language with the associated inference engine also written in FORTRAN.

Given a set of fully specified gear wheels, their speed of rotation and the power transmitted as well as some further design information, COPART will completely detail design a shaft for those gears. This process involves the selection of bearings and the dimensions of the shaft as well as checking such varied constraints as the stresses and strains, and whether every component can indeed be mounted on the shaft.

In COPART the gear shafts are described as a list of sections which may be either conical or cylindrical and include further attributes describing the nature of the surface, for example, whether it is carrying a gear wheel, a key way, or some type of bearing. Geometric reasoning about, for example, whether the gears and bearings can be assembled onto the shaft, can then be expressed solely in terms of the list of sections such as the need for the diameters to decrease up to the end of the shaft.

On the other hand, given a power transmitting system such as a set of gears and their service conditions, the CARTER system will design a casing for it which meets certain constraints while attempting to minimise some cost function which is currently the weight of the casing.

Unlike COPART, the CARTER system truly develops the geometry of its solution through use of ingenious algorithms for the initial layout of the overall shape followed by finite element analyses both to determine the need for stiffness improving ribs (which might be considered as the use of a "design feature") and to finalise all the wall thicknesses.

Both systems were developed in an engineering environment and most of the emphasis in the publications is placed on the central engineering calculations, as opposed to the guidance of the design process.

2.3.3.2 The AIRCYL system

The AIRCYL [BROW86, BROW85a, BROW85b, BROW85c] system's area of expertise is the design of pneumatic actuators. The LISP language was used to write an interpreter for the knowledge representation language used in AIRCYL. Of all the Knowledge Based Expert Systems for design, AIRCYL is the one which is described in greatest detail.

The knowledge of the system is organised in a hierarchical manner in order to manage the complexity of the design process. Each element in this hierarchy is called a "specialist" and is responsible for some increasingly more detailed section of the component being designed. To this end it calls several lower level specialists in a prescribed order corresponding to a small design "plan". At the bottom of this hierarchy individual design steps are performed, for example, to determine the value of a specific design parameter. Constraints which test for particular relationships between several design parameters occur throughout the hierarchy. In order to recover from a constraint violation each set of design steps or specialists will in fact have several recovery measures or alternative "plans" that can be brought into effect.

The design activity proceeds in four stages. First the design requirements are obtained from the user. Next the system enters a rough design phase in order to determine those values upon which much of the subsequent design depends. This has the positive side effect of reducing the subsequent search space. At this stage provision has also been made to overcome the problem of mutually dependent design parameters which would otherwise prevent the design from progressing. The system will select a value for one of the parameters and thus break the mutual dependency. The third phase is the design process itself and is essentially top down refinement with each level of the hierarchy adding to the design and checking any relevant constraints. The final stage is one of redesign and only

occurs if a failure (that violated a constraint) occurred. This is again handled hierarchically, but this time the hierarchy is used in the opposite direction, that is the attempt to overcome the failure will start at the (low) level where the failure first occurred. Only when all possibilities at that level failed to resolve the problem, will the failure be reported to the next highest level in the hierarchy. When the problem is finally resolved, the design stage will resume from where it left off.

2.3.3.3 The ADROIT system

The ADROIT system [ALSI86a, ALSI86b] is an ongoing project to assist in the design of airliners and in particular capture some design knowledge in both the area of wing design and the global aircraft configuration which involves deciding the position and shape of the main elements of the aircraft, such as the fuselage, the wings, engines and tail fin.

Although the ADROIT system mainly concentrates on the details of the compiled aeronautical engineering knowledge, its structure appears to be broadly similar to the AIRCYL program, with the significant difference that it is mainly meant to be used interactively with the designer exploring several design alternatives while making use of the knowledge base to assist him in this task.

ADROIT is being implemented in PROLOG.

2.3.3.4 The PRIDE system

The PRIDE [MITT86] system operates in the area of the design of paper transport systems within photocopying machines.

In the PRIDE system the design process is considered as a knowledge guided search through a large and complex space of possible designs, in association with analysis and constraint checking and thus uses a “generate, test, analyse, advise, modify paradigm”.

The PRIDE system is highly flexible as it can operate both interactively (with the human both guiding some of the activities and making changes to the current solution outline) and as an autonomous knowledge based design system.

The structure of the knowledge within PRIDE is fairly similar to that of the AIR-CYL program. Design “goals” both decompose the process of design into simpler steps as in an abstract refinement hierarchy and form key concepts around which all design knowledge can be organised.

The main difference between PRIDE and the other systems appears to be the consequence of the implementation language. PRIDE is implemented in LOOPS which is a multiparadigm object oriented knowledge programming environment implemented as an extension to LISP. The object orientation of LOOPS allows a sophisticated hierarchy of *classes* to be devised for storing the different types of design knowledge. This gives the PRIDE the advantage of an explicit modular and maintainable description of the *format* within which its knowledge is kept and contributes substantially to the ease with which the whole system can be constructed, extended and maintained.

The common feature of all the above systems is that they are constructed around a very specific, very well defined type of design task, and exclusively operate within that area of speciality.

Furthermore, their “expert knowledge” is constructed around a specialised, parametric representation of the constituents of any design that the given system can

produce. During design itself, the expert knowledge is principally used to guide and organise the computations and constraint verifications.

This structure does not reflect any of the design process steps suggested by the design methodologies (see Section 2.1). Rather, the design progresses as though all of the abstraction and functional decomposition activities had already occurred and thus reduced the remaining design task to finding a path through the (mainly numerical) details of the fixed constraint space designated by the current specification [MOST85].

As a consequence the creativity of the system is restricted to selecting a combination of predefined parametric answers and allocating values to the parameters. In particular, this approach thus avoids having to deal with the details of the decision making process (see Section 2.1).

This approach to design is clearly not universally applicable as the prerequisite is a sufficiently small and well understood design task to be reducible to a set of rules, and is thus primarily only suited to some "routine" design tasks [BROW86].

When considering an area in which to apply such a system the first problem is to devise of a scheme capable of representing all (parametric) design solutions that it will ever be able to produce. The "expert knowledge" can then be extracted and organised so as gradually to narrow down the set of possible answers and be structured into individual calculation modules, whose formulation tends to be dominated and determined by the way in which the potential "design solutions" are represented in the first place.

A further problem is that, in the absence of any computerised common sense, every single compatibility constraint (for example that each hole must be smaller than its containing block) must be specified in terms of the detailed parametric design solutions, otherwise the system will, in certain circumstances, fail to produce a sensible answer.

Restricting the area of expertise reduces all these problems. By selecting a suitable type of design task both the representation and the geometric (and

other) constraints problem can be overcome by a specialised data structure, which can hold all design aspects in a suitable form both for decision making and constraint enforcement, as illustrated by the above systems.

Some of the above systems (COPART, CARTER, AIRCYL) reflect a tendency towards fully automatic design : a specification is input into the system and the system automatically outputs a finished design. Therefore these systems are not destined to give the designer a very profound insight into the design problem. Instead, the purpose of these systems is much more one of alleviating the burden of "routine" design tasks. When the system provides more interactivity (PRIDE, ADROIT) the designer can explore, and develop a better understanding of, the design problems within the bounds of the systems knowledge base. However, it remains to be seen to what extent this improvement in implementation technique renders the current approach applicable to more complex design tasks where there is greater variation amongst the design solutions and the number of alternatives to be considered is correspondingly larger.

The additional benefits of these systems mainly result from their representation and use of knowledge. The bringing together the expertise from multiple experts and different sources within a same formalised knowledge base has the advantage of both clarifying and rendering explicit information which might otherwise be difficult to access. Furthermore, the computer system will ensure that any piece of "knowledge", once formulated, will be used systematically in all future designs.

The next Section will consider a few systems which propose different ways in which to support design activities.

2.3.4 Other Approaches

It is clearly unrealistic to believe that all developments fall within the spectrum described in the above Sections. This will now be illustrated by exploring four very different ideas and systems for design.

The first of these is a radically different view of the design process as an optimisation task [BOHA85]. This means that design is the process of finding the global extremum of a “function” (called the design mapping) which links the specified design performance and the space of all design configurations.

The difficulties with this view are that in most practical applications the two spaces linked by the design mapping are multi-dimensional, and a figure of 100 to 200 dimensions is said not to be unusual [BOHA85]. Furthermore the design mapping cannot, in general, be represented explicitly. Indeed, it might be argued that the whole purpose of the design methodologies outlined in Section 2.1 can be seen as a gradual and systematic exploration of the most promising parts of this mapping. Finally, the design mapping is not a one to one mapping : not only does it not cover the whole of the configuration space (as there are design configurations which do not exhibit any meaningful behaviour) but it is also not even a “function” in the mathematical sense as there are usually several design configurations which will exhibit the same or a very similar kind of performance.

Assuming that all these problems can be overcome, (as in the case of the illustrative design examples presented in [BOHA85, HOWE86]) the design task becomes one of finding the global extremum of a multi-dimensional “function” with numerous local extrema which may not be everywhere differentiable. However this can be achieved by means of a generalised “hill climbing” algorithm [HOWE86] or a “simulated annealing” algorithm [BOHA85] which is capable of escaping from local extrema in favour of the global extremum.

The next system, ROSALIE [CHOL83a, CHOL83b], is concerned with the management of design relationships. In essence, ROSALIE is a database development. Its originality lies in that it allows the user to manage not only the design data itself but also “relations” which should hold true amongst this data. The

designer can thus not only edit design data but also the kinds of relationships (expressed in PROLOG) that should exist between them, and, at any stage during the design process the status of these relations (that is, whether they hold or not) can be determined. Any defined relation can further be augmented by a program which will alter the design data so as re-establish the relationship whenever it is violated. Unfortunately, the system does not seem to have progressed beyond the initial prototype stage (for example the user has no control over the automatic propagation of changes) and cannot be considered as a CAD system as there is no graphics interface. However it does allow a designer to explore complex sets of interrelationships and thus provides assistance with the preliminary information gathering stage of design decision making.

The next system is also concerned with design relationships, but only those of a numerical kind. The DESIGNER system [MACC85] is a knowledge based assistant for modelling the preliminary numerical inter-relationships in the early design stages. Numerical relationships between design variables are represented as a directed network, with the nodes as the numeric variables and the links as influences or dependencies. The system can provide the designer with estimates of values, update existing values when a change is effected, deduce the strength of the influence of one parameter on any other and explain how it reached any of its answers. The system relies on the designer for controlling the design process and uses its built in expertise to search through the relationships and deduce influences and the strengths of influences amongst the numerical parameters. However, in its current form, the system does not contain any knowledge for dealing with spatial arrangements. The assistance provided to the designer is essentially the same as that of the ROSALIE system above but more particularly targeted towards numerical relationships.

The final system, DACON [RUNC84, SWIF85, SWIF83], is also knowledge based, but operates as a consulting system giving advice on the suitability of a component with an axis of rotational symmetry for processing by automatic assembly machinery. The rotational symmetry of the component is important as it allows its geometry to be fully described by a two-dimensional cross section. The two-dimensional section can further be represented by a list of nodes joined

by straight line sections (although an extension to more complex shapes such as arcs does not, a priori, appear to be unduly complex). As lists are a well understood data structure in Artificial Intelligence, this provides an elegant way of incorporating design geometry into the knowledge based system. In particular, the list representation is available directly in the PROLOG programming language which was used for building the DACON system. The list can then be manipulated using the powerful list manipulation facilities offered by PROLOG so as to extract higher level information such as whether the component has end to end symmetry by checking the list for such features as whether the coordinate values are palindromic or not. This high level information can then be used directly to infer, for example, how easily the part can be oriented in automatic part feeding systems. The general rule would be, for example, that parts which present end to end dissymmetry are easier to handle. The system can even suggest one of its small, pre-defined design changes (for example the addition of a step at one end of the component) to break the symmetry of a part and so improve its orientability. The support given to the designer is essentially that of a comprehensive source of knowledge which automatically applies those parts of its information which are relevant to the current design. In this particular case, the help is concentrated on the *evaluation* of a design with respect to the assemblability criterion during the decision making process (see Section 2.1).

2.3.5 Summary

A variety of ideas for advanced computer aids to design have been discovered. In each case an attempt was made to related them to design theory and whenever possible the representation of spatial and geometric aspects were described.

The advanced systems for design are mainly concentrated on

1. automating small and well understood "routine" design tasks in the form of design elements (SICAM, JIGS) and, even more complex "routine" design tasks (which have to very well understood indeed) in the form of whole knowledge based expert systems (COPART, CARTER, AIRCYL).
2. helping the designer to explore a complex design space resulting from either a set of interdependent relationships (DESIGNER, ROSALIE, ADROIT) or obtained through the compilation of knowledge from many different sources (PRIDE, ADROIT).
3. providing the designer with detailed advice (DACON).
4. providing evidence that reasoning about geometry is but a particular case of reasoning about real valued quantities (HIMADES).
5. improving the representation of and interaction with *geometric features*.

However, there seem to be few attempts at modelling the kinds of information processing activities suggested by design theory and methodology. Nevertheless, such activities are needed prior to building the kind of knowledge based systems described above (in order to create the level of understanding needed for such an enterprise) and further to guide and benefit from the exploration of complex design spaces.

This therefore leaves many open questions such as, for example, how can the relationships (see Section 2.1.2) between design goals be inferred ? how to select between strategies for dealing with a goal when there are several to choose from ? etc [MOST85].

Another important problem is the meaningful integration of geometry into such knowledge based systems. Although general purpose and informationally complete schemes for representing the geometry of engineering components already exist in the form of solid modelling systems, the way in which that information is represented is not suited for the kind of decision making required during design [LEE.87, MILL85]. For this reason, CAD systems will display pictures of the modelled component rather than a printed list of, for example, its CSG tree. The picture allows the designer to reformulate and use only those parts of the pictorial information that are relevant to his current concerns in a way which the printed list does not. While the mechanisms of this information handling process remain unknown, the integration of freely specifiable geometry into expert systems will remain as a question without a definite answer. Particularly as any system based on some form of taxonomic organisation of design concepts will encounter problems [POPP86]. These occur whenever some design element combines the functions of two or more other elements, or when it can serve different purposes and thus requires different representations with the emphasis in each placed on those features which are relevant to that application.

Thus one direction of development for advanced design systems lies in the representation of design concepts and geometry associated with guidelines on how to use them.

The next Section will, however, first investigate qualitative reasoning about real valued quantities.

2.4 QUALITATIVE REASONING

One of the aims of qualitative reasoning research is to develop models of human common sense reasoning applied to the physical world [DEKL84, FORB84, KUIP84], in particular, for use in future knowledge based systems.

The problem with using differential equations as models of physical phenomena is the computational cost of both solving them and then obtaining a meaningful interpretation of the answer.

If numeric computation methods are used for the solution process, large amounts of computer resources will be needed and the ensuing interpretation process is both complex and does not reveal the nature of the dependencies between different parameters, nor how the outcome might vary for a different set of initial values. On the other hand, the search for an analytic solution requires global and knowledge intensive operations such as indefinite integration, and may often fail to produce a closed form solution.

The difference between these approaches lies in the representation of continuously varying parameters. In the first case they are treated as real numbers and in the second as real valued continuous functions. Treating them as a real number results in a very weak model as the only pertinent information is the value of the number which either is, or is not, known (in this latter case the solution cannot be computed). Treating them symbolically as real valued continuous functions gives access to a very rich set of symbolic relationships (such as the arithmetic, trigonometric logarithmic or power functions) for describing both the problem and the solution.

A qualitative reasoning method should, on the contrary, be able to handle imprecise descriptions of physical systems such as weakly specified functional relationships or non numeric specification of parameter values. It should further be able to cope with systems which do not have closed form solutions and identify unexpected points of qualitative change while not requiring extensive computational resources [KUIP84].

These are all properties which also make qualitative reasoning interesting for design. In particular the ability to both represent and obtain deductions from descriptions which are but outlines of a system should be valuable for those stages of the design process where complete descriptions are not yet available and decisions must nevertheless be made.

The two main problems which have to be addressed by qualitative reasoning are the question of *representation* of real world physical phenomena and behaviours and the *limits to the deductive power* that such descriptions can support. As qualitative reasoning appears to be a relatively new field of artificial intelligence slightly different approaches are still being explored in parallel. This is illustrated below by the ENVISION [DEKL84], GIZMO [FORB84] and QSIM [KUIP84, KUIP85] systems. In particular the following Sections deal with current qualitative representation and reasoning schemes as well as providing an extremely brief look at future research (needs).

2.4.1 Qualitative Representation

There are three aspects which need representing qualitatively, namely real numbers, physical systems and the behaviour of physical systems. These will be explored in that order.

The most fundamental issue of a qualitative reasoning system is its representation of continuously variable parameters.

The method used in the ENVISION system [DEKL84] is to allow each variable to take only a limited number of symbolic values such as “-”, “0”, “+”. These are typically interpreted as “below some threshold value” or “decreasing”, “equal to a threshold value” or “not changing” and “above some threshold value” or “increasing”. The GIZMO system [FORB84] uses a more comprehensive representation of a parameter by storing its value (that is both the sign and the magnitude) as well as the value (that is again both the sign and the magnitude) of its first derivative. However, in practice GIZMO actually uses a qualitative

description based on the (partial) ordering between parameters rather than performing numeric computations. Finally the QSIM system [KUIP84, KUIP85], uses a similar but slightly simpler representation scheme where parameters are only described in terms of ordinal relations and directions of change.

The next level of description involves representing the structure or function of the device to be modelled.

The ENVISION system is based on the formalisms of linear systems theory using descriptions in terms of lumped ideal elements which interact through ideal interconnections. Thus a device is modelled as a graph in which the nodes represent *components* and the edges represent *conduits* along which there is a flow of some *material*. The behaviour in conduits is described in terms of a pair of attributes, one pressure like and the other flow like (for example, velocity and force, or voltage and current). The behaviour of components is modelled by sets of differential equations (in qualitative variables) corresponding to different operating regions themselves defined by inequalities amongst the qualitative variables.

On the other hand, the GIZMO system is based on modelling of the *processes* of change rather than objects or devices. In fact objects are represented simply as a set of parameters. A process of change is something that acts through time to alter some parameters and occurs whenever certain conditions are fulfilled. These conditions consist of the existence of the appropriate types of objects (and their parameters) and two kinds of preconditions that the objects must meet, namely those that can be represented within the qualitative theory and those that cannot. A process consists of a set of relations that it will impose (when it, the process, occurs) amongst the parameters of the concerned objects, any new entities that are created and a set of "influences", that is, causes for change in object parameters. A language for combining processes into larger more abstract entities is also provided.

Finally the QSIM system concentrates directly on sets of qualitative differential equations and does not represent either devices or processes directly. The qualitative differential equations consist of three principal types of constraining

equations. These are arithmetic ($X = Y + Z$), derivative ($Y = dX/dt$) and proportional ($Y = M^+(X)$ meaning that Y is a strictly increasing function of X). The proportionality relationship, in particular, provides a very much weaker level of description than would be possible with a numerical or analytic approach to the solution of differential equations.

Because of the qualitative nature of the modelling, a given qualitative description will characterise different physical realisations of the same device [DEKL84]. This feature contributes to making qualitative representation methods highly attractive for design problems as it means that alternative avenues of design development can be evaluated at an early stage without first having to pursue each to a (potentially wasted) level of detail.

The converse is also true; that is, a given system can have different qualitative representations [KUIP84] which reflect different levels of detail, and thus may not all be appropriate to the current problem.

Finally, the results of the qualitative reasoning must be represented.

ENVISION represents the modelled device as always being in some state. Each state is uniquely identified through a set of qualitative values for the parameters and may last for a finite time or be only instantaneous (as for example, when a parameter passes through some reference value). The states are themselves linked in a directed graph. Any path through the graph corresponds to a possible behaviour of the device and will depend on the current state of input to the device. The whole graph is therefore a complete description of all possible behaviours of the device.

In QSIM the behaviour is described using a finite set of qualitative time points which again represent the qualitatively distinct states of the system. Again each qualitative time point corresponds to either a point or an open interval in real physical time. At each qualitative time point the values of the parameters are described in terms of ordinal relations and directions of change. During an open time interval, the physical system is in a state of continuous change during which

the qualitative parameter values remain constant and the behaviour therefore remains within the scope of the same qualitative description.

In the GIZMO system this representation is augmented by the history of what processes are occurring during the qualitative time points.

2.4.2 Qualitative Reasoning

The objective of qualitative reasoning systems is essentially to derive the behaviour of an arbitrary device solely from its qualitative model.

For example, the input to the ENVISION system consists of a set of components and their allowable paths of interaction (the device's topology), the input signals applied to the device (if any), and a set of boundary conditions which constrain the device's behaviour. ENVISION then produces the description of the corresponding behaviour.

The GIZMO system works more in terms of finding all possible processes, determining whether these are active (that is, whether their preconditions are fulfilled), computing the changes in parameter values, and performing a kind of limit analysis to deduce the pattern of beginning or terminating processes.

In QSIM the qualitative reasoning proceeds in two stages. The first is to complete the description of the current state by propagating the implications of initial facts through the qualitative equations at the current time point. The second predicts the characteristics of the next distinct qualitative state, which may involve deducing the existence of a previously unsuspected limit value. If the qualitative deduction process becomes intractable the qualitative structure description can be summarised and simplified [KUIP84]. The new description is less detailed and therefore weaker than the old one, but because it is simpler, it may avoid the problem and thus allow the qualitative deduction to proceed.

In the case of the QSIM system, it has further been proved that its qualitative deduction algorithm will produce every actual behaviour of the device being modelled.

However, because of its local point of view, it will also predict spurious behaviours which will not be exhibited by any mechanism satisfying the structural description [KUIP85].

This is because qualitative reasoning allows states of partial knowledge through using a weak set of primitive relations and operating on symbolic (qualitative) descriptions of real numbers rather than the real numbers themselves.

As a result the proposed qualitative structural and behavioral descriptions only require a simple deduction process based on the ability to create and match simple expressions, rather than on arithmetic operations or symbolic integration. Partial knowledge can be represented in the sense that both parameter values and functional relationships are only constrained to lie in qualitatively defined classes rather than being described explicitly.

However, when defining the qualitative law of addition there are certain cases where the qualitative information is insufficient to uniquely determine the outcome. In particular the addition of a value qualitatively described as being in the interval $(0, +\infty)$ and one in the interval $(-\infty, 0)$ is not deterministic and may result in a value which is either in either one of the intervals or is zero.

The consequence is that qualitative reasoning sometimes underdetermines the behaviour of a system and that a set of possible behaviours has to be produced, only one of which is actually exhibited by the physical system being modelled. All the above systems suffer from this problem. As they all operate exclusively with qualitative descriptions, and as no working system has as yet been reported which uses both both qualitative and quantitative descriptions it is not known how easily this problem can be overcome.

2.4.3 Future

Qualitative reasoning techniques are being used within larger projects [CHAN85]. Two of the more interesting projects are PROMPT [ADDA85] and EDISON [DYER86]. The PROMPT system “is aimed at producing solutions to tasks such as designing a pen that can write upside down; why is my pen not writing on glass ?; What will happen to a pen at 200 degrees ?” [ADDA85] and proposes to investigate a representation scheme for “complex physical domains” but cannot yet cope with the representation of shape. Of an essentially similar nature, the purpose of the EDISON [DYER86] project is to experiment computationally with processes of invention, analogy, and naive mechanical device representation. Devices are represented symbolically and reasoned about without performing numerical computations. Essentially this involves synthesising a taxonomy which is sufficiently detailed for the stated goals of the project.

Because qualitative reasoning is a relatively recent branch of research there are still some problems which are not yet resolved. These are concerned with the mechanism of qualitative deduction itself but also raise much larger issues.

The problem of underdetermined results, that is the theoretical issue of how many qualitative equations determine a unique solution (particularly as n independent simultaneous qualitative equations in n variables do not necessarily have a unique solution [DEKL84]) has already been mentioned above.

The larger issue is that of problem formulation, or in this case that of selecting which of several ways a physical situation should be described to allow a deeper examination [DEKL84, KUIP84].

2.5 DESIGN INFORMATION SUMMARY

So far this chapter has considered the areas of design theory, computer aided design, advanced systems for design and qualitative reasoning. The common thread to all these areas of research is that they are relevant to the search for an understanding of the kind of information processing that is involved in design. Some emphasis was placed on the representation of information and, in particular, of geometry as this is central to mechanical engineering design. The processes of reasoning about geometry are particularly challenging because although they are performed with moderate ease by humans, they remain beyond the capabilities of computer systems. This is in spite of solid modelling systems which do maintain informationally complete descriptions of shapes.

The remainder of this Section will propose an explanation of why this should be the case, that is, if geometric (solid) modelling systems are sufficiently powerful to meet most geometric-representational needs, why cannot design systems take full advantage of this power ?

To understand the problems involved, it is necessary to appreciate the nature of the different kinds of *information* involved in solid modelling and knowledge based expert design systems. Two relatively recent concepts from the field of Artificial Intelligence can be used. These are the distinction between *surface* and *deep* knowledge. As these concepts do not always seem to have a universally accepted interpretation the distinction they introduce will first be illustrated by a medical analogy. Surface knowledge consists in linking observed symptoms to known diseases. Deep knowledge tends to be more oriented according to specific disciplines, such as anatomy or biochemistry.

Surface knowledge is clearly application specific and domain dependent whereas deep knowledge is generic to a discipline and can be used in different application areas. The ability to be of use in different situations is precisely what makes deep knowledge so valuable.

In the context of design, surface knowledge would guide the design process, and deals with the question of what to do next ? whereas deep knowledge would be

used for analysing aspects of the design and consists of knowledge from disciplines such as geometry (for, say, computing intersections), kinematics (for deducing the motion of a mechanism), etc.

As a task, engineering design clearly requires both kinds of knowledge.

Systems which deal with each kind of knowledge individually work satisfactorily. For example, in the case of surface knowledge one of the popular, general purpose, systems is the programming language PROLOG [CLOC81], which with a few enhancements, can be used to form a shell of a knowledge based system (for example, APES, the Augmented Prolog for Expert Systems [HAMM87]), and in the case of deep (geometric) knowledge there are already many working solid modelling CAD systems in use in industry.

However, problems arise when attempting to combine deep and surface systems. These are the consequence of a difference in the representation of information. Knowledge based systems are based on the fundamental concept of defining and exploring in a goal directed manner a large search space of distinct *discrete* states of knowledge. For example the knowledge programming language PROLOG uses the Horn Clause subset of first order predicate logic to define a search space which is then searched in a systematic manner by a depth first deduction system using clause unification and resolution. On the other hand, geometric modelling systems and engineering analysis packages are concerned with *continuously* varying parameters represented by floating point arithmetic. The essential difference is that the first kind of system operates on discrete symbols whereas the second has to cope with the continuity of the real numbers.

Combining these two kinds of system therefore necessarily involves dividing the continuous quantities into a finite set of alternative concepts which can then be represented symbolically. This operation can either occur when the combined system is built or be postponed and only occur at run-time when the system is actually solving a problem.

Dividing the continuous quantities into symbolic pieces when the combined system is being built effectively hides the deep system by providing the surface

system with a discrete set of *specialists* which use the deep system to arrive at their deductions. The idea is similar to that of the built in predicates in, say, PROLOG for certain tasks (as, for example, floating point arithmetic) which would otherwise be exceedingly difficult to achieve directly within the confines of the surface system.

This approach has been used in an assembly planning system [JAKO85b] which contains a deep knowledge kinematics "specialist" capable of deducing the instantaneous kinematics of a mechanism from its solid model description provided that the links have surface to surface contact (as opposed to line or point contacts) and that the surfaces are either planar, cylindrical or spherical. Other kinds of contacts or surfaces would require providing the system with more complex "specialists".

The drawback of this method is therefore that the deep knowledge is only accessible if a suitable set of "specialists" was provided when the system was built. This raises the question whether providing such a predefined interface between the deep and the surface system does not prevent the deep knowledge from being of use in circumstances which *were not anticipated* during the construction of the combined system, a weakness which would sensibly reduce the advantage of having the deep system in the first place. The knowledge based systems for design considered above (Section 2.3) resolved this problem through selecting a sufficiently narrow field of expertise which clearly reduces the magnitude of this problem.

On the other hand, the very great advantage of this approach is that the surface system remains essentially unmodified. This means that the construction of the knowledge based design system is simplified as an already available deduction system can be used.

Building appropriate "specialists" into combined surface and deep system essentially assumes that dividing the continuous (deep) geometric quantities into a finite set of alternative symbolic concepts can be suitably done in advance.

The alternative approach is to delay the process of dividing the continuous (deep) geometric quantities into a finite set of alternative symbolic concepts until run-time where it could be tailored to the current needs along the lines proposed by qualitative reasoning techniques.

This clearly requires changes in the surface reasoning system in order to accommodate this additional task and should take into account the additional properties of continuously varying parameters including the kinds of information that can be expressed through them and how they are used in practice.

In this context, the consider following situation. A simple description of the kinematics of a mechanism may consist of a series of graphs describing how the position (typically expressed in terms of translations along coordinate axes, and rotations about them) of each link changes as the mechanism moves through its operating range. Although these graphs are not "solid models" themselves, they are totally determined by the geometry of the links and their connections.

The process of deriving the kinematic description of the motion from the solid model (or, more often, from a simpler kinematic joint model) is one of numeric computation.

The designer will, however, not use all of this information, but only verify some critical values. If these are not satisfactory the mechanism is adjusted and the kinematic motion description recalculated. This iterative loop may then be repeated as often as necessary.

A deep, geometric system would calculate the kinematic displacement of each point on the mechanism up to an unnecessary degree of precision, whereas a purely symbolic system might, on the other hand, not provide sufficient detail where the designer needs it.

An *ideal* system would be capable of quickly computing a representative (symbolic - discrete) approximation to the kinematic displacement curves and of refining these in the areas of interest to the designer to the extent that he requires (or his model permits), using more and more deep (numeric) computations.

The structure, nature and implications of such an *ideal* system will be considered in the next chapter. However, there is first a final, important point which must be clarified.

This is concerned with the relationships between geometric entities, continuously varying parameters and qualitative representations. Qualitative representation systems described above (Section 2.4) only apply to individual, that is one-dimensional, continuously varying parameters whereas geometric shapes consist of higher (typically three) dimensional continuously varying quantities. This means that qualitative representations cannot be applied directly to geometric entities.

However, geometric entities (such as points, lines, arcs, surface patches, etc) can be used to represent information which is non geometrical, such as for example the graph of (a kinematic displacement) function provided that the geometric modelling system allows such an interpretation to be placed on its models.

Therefore higher-dimensional continuously varying quantities are the *generalised form of information* of which (three-dimensional) geometric shape and the (one-dimensional) continuously varying parameters of qualitative reasoning are particular instances. Any information which is expressed as a particular instance of such continuously varying quantities will hereafter be referred to as *geometric knowledge*.

Using the above kinematic example, it follows that the simple integration of an existing solid modelling CAD system and a knowledge based deduction system will prove inadequate for the representation of non-geometric, though geometry related, continuously varying data such as the description of the kinematics of a mechanism.

Thus, to sum up, the idea of a (higher-dimensional) continuously varying parameter is the fundamental concept which pervades much of engineering information and thus singling out three-dimensional geometry or one-dimensional continuously varying parameters does not make sense in a design context. Instead the concept of *geometric knowledge*, as defined above, should be used.

Chapter 3. THE GEOMETRIC REPRESENTATION OF KNOWLEDGE

3.1 OBJECTIVES AND OVERVIEW

The principal ideas extracted from the evidence gathered in the previous chapter are that :

1. Engineering design has to deal with quantities and concepts best represented by continuously varying parameters in n -dimensions, that is with geometric knowledge.
2. Artificial Intelligence has discovered that the most fundamental process of intelligent reasoning can be successfully represented as a process of searching (that is systematically trying out all potential combinations in) a large space of discrete alternatives until a solution is found.
3. Reconciling these two different views in a productive manner is difficult.

As chapter 2 illustrated, systems have been developed for each type of knowledge in isolation. However, for a quantum improvement in advanced design aids a combined system, one that incorporated the representation of *geometric knowledge* would clearly be necessary.

In order to obtain an insight into the configuration of such a system it is instructive to evaluate the knowledge representation approach taken in Artificial Intelligence from to the point of view of geometric solid modelling and vice versa. As it is the *approaches* to knowledge representation that are being compared it is not really necessary to select a pair of specific systems. Nevertheless, the comparison will be loosely based on the Horn Clause subset of first order predicate calculus

(as found in a knowledge programming language like PROLOG [CLOC81]) and a conventional (CSG or analytic) Solid Modelling system.

When considering the knowledge representation scheme of Horn Clauses for the purposes of solid modelling the following correspondences would intuitively appear to be the "most natural". An n -ary predicate would correspond to an n -dimensional space of say Euclidean geometry. The assertion of a fact using that predicate (that is a Horn Clause with an empty conditional part) would be interpreted as specifying a point. This point would have as coordinates the predicate's n -parameters and exist in the space named after and defined by the predicate. A rule (that is a Horn Clause whose conditional part consists of the conjunction of further predicates) would specify a geometric figure which can best be described as the analogue of a linear space in Euclidean geometry. Indeed if the predicate contains a variable as its n th argument then the geometric equivalent is a point whose n th coordinate is unknown and may thus range over the whole set of real numbers and therefore lie anywhere on the line (that is, the one-dimensional linear space) which is parallel to the n th coordinate axis. Similarly, if the predicate at the head of the Horn Clause contains two variables, the point specified by it will only be constrained to lie in a plane (that is a two-dimensional linear space) defined by the two coordinate axes corresponding to the variables arity position in the predicate. Executing a query (that is specifying a Horn Clause only consisting of a conditional part) will look for (all) "points" known to the system which "intersect" the "linear space" specified by the query.

The most unusual feature from the solid modelling point of view is that the whole system works without a metric (that is the "points" only have a tuple of symbolic names as coordinates), is discrete and works in any finite dimension (whereas solid modelling operates exclusively in three dimensions).

The fact that many of the "points" are not expressed explicitly but are re-deduced by the rules when needed is similar to solid modelling where only a few key points are stored for each volume patch in an analytic solid modelling system (see Section 2.2).

Conversely, when considering the knowledge representation scheme of solid modelling for the purposes of logical deduction the following correspondences would intuitively appear to be the “most natural” — these are essentially the same correspondences as used above, except that now the flow of the analogy will proceed in the opposite direction. In particular, an n -ary predicate was considered to be the analogue of an n -dimensional space. As the solid modeller operates exclusively in three-dimensional space, there would appear to be only a single three-ary predicate. Similarly, the assertion of a fact was considered as the analogue of a point within the predicate-space. Therefore, each point in the solid modeller corresponds to the assertion of a fact under the single three-ary predicate. However, a whole (non-degenerate) solid model consists of an infinite number of points. Therefore a (non-degenerate) solid model would be the analogue of asserting an infinite number of facts in an AI deduction system. Finally, in the above analogy, ‘rules’ involved the representation of whole linear spaces and correlating points within these across different predicates of possibly different arity. Such ‘rules’ cannot be represented at all in the solid modelling system. There are two reasons for this : Firstly the solid modelling system is restricted to a single three-ary predicate (ie three-dimensional space) and thus cannot express relationships between different predicates, ie spaces (of possibly different dimensionality). Secondly solid modelling is restricted to finite objects and thus cannot represent a whole, non trivial, linear space.

Viewed as a deduction mechanism solid modelling systems reflect these representational features. The central deduction operation consists of computing the intersection of solid models. Interpreting a solid model as the assertion of (an infinite number of) facts this “deduction” process consists of matching sets of “facts” (essentially by computing the intersections of the corresponding solid models). In the absence of (the ability to represent) “rules” this is the only form of deduction that can be performed by a solid modelling system.

However, the ability of asserting an infinite number of “facts” under a predicate (that is, to represent a solid model containing an infinite number of points) has a very profound consequence : the approach to deduction consisting of systematically trying out all explicit combinations will never terminate, unless some

additional mechanism (related in some way to the mechanism of intersection computations of solid modelling) is added to deal with them.

The criteria for a knowledge representation scheme capable of both kinds of information can now be easily obtained as the union of the properties of the systems dealing with each kind individually.

But first consider a further idea which was identified in chapter 2. This is that some engineering data, although not directly geometric, can nevertheless be represented by means of geometric entities. The graphs of continuous functions in a real number space (of appropriate dimensions) all lie within this category. Such graphs are used throughout engineering to represent a large variety of information from kinematics, stress analysis, vibration frequencies and a host of other "performance" data. The process of design involves using such data and making deductions from it. These kinds of deductions are different from those used with geometric solid models but are of a similar nature to those in qualitative reasoning.

All the above ideas can now be combined into an informal statement of the objectives of a system for the geometric representation of knowledge. Such a system should :

1. be able to represent geometric knowledge, that is, continuously varying parameters in n -dimensions. This involves expanding the capabilities of solid modelling to allow multiple, higher-dimensional spaces with individual models which may contain a proper linear space (and thus extend infinitely in some direction).
2. be capable of performing the appropriate kind of "deductions" on these entities, in a way analogous to the operations performed by AI or solid modelling or qualitative reasoning systems.

The transition from the informal objectives described above to the description of a system capable of being implemented forms the remainder of this chapter.

The proposed approach simply consists of five main ideas whose combination into a single system should be a first step towards fulfilling the above objectives.

The incremental introduction of these ideas commences in Section 3.2 with the description of a scheme for a generalised polyhedral modeller. This is augmented in Section 3.3 by the idea of *excursion boxes* as first suggested by Medland [MEDL86]. A mechanism for extending the generalised polyhedral modeller to represent non-polyhedral shapes is then suggested in Section 3.4, while Section 3.5 clarifies the different interpretations that may be placed on some geometrically represented piece of knowledge. Section 3.6 then returns to qualitative representations, before Section 3.7 finally provides a summary.

The importance of Section 3.5 should be emphasised as it exposes the mechanism by which geometric models may represent information concerning continuous relationships and thereby introduce a geometric approach to the representation of knowledge.

3.2 GENERALISED POLYHEDRAL MODELLER

This Section introduces the generalised polyhedral modeller which forms the kernel of the system for the representation of geometric knowledge. The system can be considered as a generalisation of a polyhedral solid modeller. This Section first introduces the two generalising steps that are involved, then examines the mathematical principles involved and finally summarises the operations that the modeller should support.

The first generalisation is to represent polyhedral models in an n -dimensional space. This is needed in situations where the geometric model does not represent the shape of an engineering component, but should be interpreted as the graph of a function, or some other non-shape related information. In this context consider the following examples.

Only two-dimensions are required to represent the graph of a function of a single variable, for example how the distance along a given path changes with time during a displacement. On the other hand, the representation of the shape of a solid object requires a three-dimensions space. The displaced positions of a moving component, for example those of a link within a mechanism observed throughout its operating cycle, consists of four-dimensional data : three dimensions are needed to represent the component in some position in space and a fourth dimension to measure how far the motion has progressed. In this case the first three dimensions correspond to the physical three-dimensional space (using units of distance) and the fourth to either time (using units of time) or the fraction of the whole motion that has already been completed (in which case no units of measurement apply). Finally, a solid object has six degrees of freedom under rigid motion transforms : three translational and three rotational. It follows that any given displacement can be described using six values corresponding to the magnitudes of the changes in those six degrees of freedom. Therefore any given displacement can be represented as a point with those six values as coordinates in a six-dimensional space. This space has been called the *configuration space* [LOZA83, LOZA81]. In a given context all points in this six-dimensional configuration space do not correspond to practical displacements because they might

bring the mobile object into collision with obstacles in its vicinity. Constructing the six-dimensional images (in the configuration space) of three-dimensional physical obstacles and computing a configuration space trajectory which avoids them is being used as a means of planning valid trajectories for robot manipulator arms [LOZA83, LOZA81].

All these examples illustrate the use of continuously varying quantities in spaces of different dimensionality to represent information (geometric knowledge) that is clearly relevant to engineering design. Therefore in order to cope with the full spectrum of such engineering information, the modelling system must be capable of representing geometric entities in any finite dimensional space with the same ease that a logical deduction system exhibits with respect to the use of n -ary predicates.

The second generalisation is the extension to the modelling of infinite objects rather than being limited to just finite ones. This feature is needed for two reasons. The first is that it effectively allows universal quantification over continuously varying parameters, which, in turn, is needed in order to express "rules" similar to those in systems based on Horn Clause logic. The second reason is that the concept of (points at) infinity is needed in qualitative reasoning systems to qualitatively describe a quantity which does not have an upper (or lower) bound by means of an open interval with one, or more, infinite end points.

The two mathematical principles involved in the above generalisations will now be explored. The n -dimensional polyhedral modeller is essentially based on the theory of *Join Geometries* [PREN79], whereas the extension to infinite objects requires some results from *General Topology* [KELL55].

3.2.1 Join Based Geometry

The main problem which needs to be addressed is to obtain a precise definition of how continuously varying parameters (in n -dimensions) can be represented in a computer. The problem centres on the fact that a (non-trivial) set of these will contain an infinite amount of elements. As the computer storage capacity is finite, these cannot be represented explicitly. This leaves the question of whether a *finite* representation of an *infinite* number of elements is a well founded proposition. That the answer to this important question is 'yes' is implicitly assumed by all existing solid modelling systems but does not seem to have been justified explicitly. (The main emphasis of earlier work was primarily placed on deriving a mathematical definition which formally captures the properties intuitively associated with 'solid' objects [REQU80a, REQU80b]). Fortunately the answer to this important question is indeed 'yes' and takes the form of the Krein-Milman Theorem which can be found in the branch of mathematics which is concerned with convex sets.

At this stage it is worth pointing out that a modern, coordinate- and (in this context more importantly) dimension-free approach to the subject of convex sets can be obtained through the systematic exploitation of the "Join" (or convex hull) operator which gives rise to the concept of "Join Geometry" [PREN79]. Further details, and more formal definitions of some of the terminology can be found in Appendix A.

A continuously varying parameter, interpreted as a geometric figure (or set of points) in some n -dimensional space, is said to be convex, when it has the following property. If any two of its points are chosen and joined by a line segment, then all the points on line segment already belong to that geometric figure; that is, no part of the line segment will lie "outside" the original figure. The convex hull of a given geometric figure, F , (again regarded as a set of points) is the smallest convex geometric figure containing F .

The Krein-Milman theorem then states that (in a Euclidean Space) a closed and bounded convex set is the closure of the 'Join' (or convex hull) of its extreme

points [EGGL69, BERG63, ROCK72 PREN79]. In other words, a (bounded and convex) polyhedron, for example, can be described completely and unambiguously by the set of its extreme points (which are more often known as vertices). If the set of extreme points is known then the convex hull operation applied to this set of extreme points will yield the desired description of a continuously varying parameter, and may be interpreted as a (convex) geometric figure in some n -dimensional space. Therefore only the actual set of extreme points need be stored. If this set is finite, then a finite description of a continuously varying parameter ranging over an infinity of points is possible.

Examples of convex geometric figures with a finite number of extreme points (vertices) are a single point (that is a zero-dimensional space), intervals on the one-dimensional real number line (note that intervals are of practical interest as they are used within one of the qualitative reasoning representation schemes, see Section 2.4), convex polygons in the (two-dimensional) plane and convex polyhedra in (three-dimensional) space. The general term for these and their higher-dimensional equivalents is that of "polytope" [PREN79, p133]. (A simple example of a convex geometric figure which has an infinite number of extreme points can be found in the form of a solid sphere where every point on its surface (infinitely many of them) is an extreme point).

The next Section will extend the representation scheme to infinite entities, a process which requires some changes to the modelling space itself.

3.2.2 Compactification of Space

The problem with the above approach is that the continuously varying parameter may only range over a bounded region (that is, a region of finite size) of space. To overcome this restriction and allow continuously varying parameters to range over an unbounded region, a more general formulation of the Krein-Milman Theorem involving the topological concept of compactness is needed as a first step. Note that in a Euclidean n -dimensional space any closed and bounded set is compact [KELL55 p144].

The more general formulation of the Krein-Milman Theorem is that each compact, non-empty convex set is identically equal to the closed join (or convex hull) of its set of extreme points [BERG66 p174].

One of the 'simplest' unbounded sets over which a continuously varying parameter may range is the set $\mathbb{R}$ of all real numbers which can also be interpreted geometrically as an unending straight line. This set is not compact. However, it can be made compact through the addition of two points, $+\infty$ and $-\infty$, and yield the (compact) extended real number line [BERG66 p71, KELL55 p149]. Note that the real number line (extended or not) is also a convex set, and in the case of the augmented real line can be described finitely as the closed convex hull of its two extreme points $+\infty$ and $-\infty$.

Thus for continuously varying parameters in one dimension the problem has been solved. The extension to higher dimensions is however fairly simple using the Tychonoff Theorem [KELL55 p143, BERG66 p83].

The Tychonoff Theorem states that the Cartesian product of a collection of compact (topological) spaces is compact (relative to the product topology).

Consider a higher, say n -dimensional, space obtained from the Cartesian product of n (compact) extended real number lines. This space will also be compact (Tychonoff Theorem) and therefore representable as the closed convex hull of its extreme points (Krein-Milman Theorem).

The theoretic foundation for the representation scheme of continuously varying parameters in n -dimensions having been successfully established, the next Section considers the kinds of primitive operations that will be needed.

3.2.3 Modelling Operations

In this Section an initial overview is obtained of the primitive operations that will be needed for the practical representation of continuously varying parameters.

For the representation scheme proposed above to work the 'Join' (or convex hull) operation must be built into the system. The Join operator must be able to compute the convex hull of a finite list of points, in n -dimensional space where each point may be either finite or at infinity. (Issues related to the implementation of this operation or any of the others will be considered in the next chapter).

For non-convex shapes the representation scheme must be extended. A very simple option is to include the ability to form lists of the primitive convex polytopes. If such a list has the appropriate mathematical properties (for example, non-overlapping, connected, bounded, closed, regular, (semi-analytic) and three-dimensional [REQU80a, REQU80b]) it may be interpreted as a non-convex solid object. The ability to form lists of polytopes (which may or may not have these properties) is not usually found in solid modellers, but is one of the most fundamental representational abilities of Artificial Intelligence deductions systems and should therefore be retained.

Like most solid modellers the system should support the basic set theoretic operations. Of these intersection and set theoretic difference are the primitives as the union operation consists of forming a list of the intersections and differences. The intersection operation on convex polytopes is closed as the intersection of convex polytopes is itself a convex polytope [PREN79 p525]. The set theoretic operations of difference and union do not always yield a convex polytope. However they will always result in a finite list of convex polytopes.

The representation scheme can support unbounded entities which includes the important category of linear spaces. Examples of linear spaces are straight

lines (one dimensional linear space), planes (two dimensional), three dimensional space, etc. These kinds of entities are sometimes found in CAD systems in the form of construction lines which, although not part of any CAD model, do provide help with the geometric construction process. The kinds of operations that are most relevant would be a linear hull operator which would create the least linear space containing its operand and a linear complement which would create the linear space orthogonal to (the linear hull) of its operand. In an n -dimensional space the linear complement to an m -dimensional linear space is the $(n - m)$ -dimensional linear space which is orthogonal to it while containing an identified point.

The remaining operations are related to defining spaces within which continuously varying parameters will reside. Apart from the facility for creating new n -dimensional spaces, operators are needed for moving information between them : these are simply the Cartesian product (for combining lower dimensional information into a higher dimensional space) and orthogonal projection (for reducing higher dimensional information into a lower dimensional space) operations.

This completes the list of the more interesting or unusual operations required by the proposed geometric knowledge representation scheme. Many other 'house keeping' commands will be needed in an actual system for such tasks as editing or copying existing data and interfacing with the permanent storage media of the computer. Any additional operations will be introduced as needed in the following Sections as additional representational methods are introduced.

3.3 EXCURSION BOXES

This Section is concerned with the concept of excursion boxes [MEDL86], their definition, properties, purpose and context.

3.3.1 Definition of Excursion Boxes

The concept of excursion boxes was first suggested by Medland [MEDL86] in an engineering context, where an excursion box is defined as the least orthogonal box, whose sides are parallel to the coordinate planes, enclosing a given CAD model of an engineering component. The dimensions of the excursion box of a particular CAD model of an engineering component correspond to the excursions of that component along the respective coordinate system axes.

In the context of a system for the representation of geometric knowledge the concept of excursion box must be generalised to higher dimensions. In fact the excursion box of any n -dimensional continuously varying parameter can be constructed by taking the Cartesian product of the (join) of the projections onto each of the coordinate axes of space. If the continuously varying parameter ranges over a connected region then the projections will always consist of a (convex) interval (or its degenerate form, a point) and the join operation is not necessary.

To understand the role of excursion boxes in the representation of geometric knowledge an insight into their properties is needed.

3.3.2 Properties of Excursion Boxes

From the definition given above, excursion boxes can clearly be both represented and constructed within the proposed representation scheme. Furthermore excursion boxes are convex polytopes as they are formed through the Cartesian product of intervals which are instances of convex sets. This together with the properties shown below give excursion boxes an intuitive appeal of 'simplicity'.

From a practical point of view, an excursion box has only limited storage requirements as it can be represented by just two points (which are such as to define a main diagonal of the excursion box), that is $2n$ numbers for an n -dimensional space. Thus storing an excursion box along side each continuously varying parameter is a feasible proposition.

The advantage of storing excursion boxes comes during intersection computations. If the excursion boxes are disjoint, so are their contents. The computational complexity of comparing two excursion boxes is low as this only involves a sequence of comparisons on the $2n$ numbers defining each box. In particular, the comparisons can be made independently along each coordinate direction (where they reduce to comparing two intervals) and then combined to yield the relationship between the excursion boxes themselves (see Appendix B). Thus when many items potentially intersect the excursion boxes can first be tested so as to quickly eliminate unnecessary intersection computations. The computational requirements of the problem of finding all intersecting pairs amongst a set of excursion boxes in n -dimensions has already been the subject of some research [EDEL81, SIX_82] which has been summarised in [LEE.84].

The greatest advantage obtainable from excursion boxes is when they are organised into a hierarchy [MEDL86]. At the lowest level in this hierarchy the excursion boxes are associated with individual polytopes. Entities consisting of lists of polytopes can themselves be represented by a larger excursion box. This larger excursion box is obtained by finding the least enclosing box of excursion boxes of the members of the list. This process of representing sets of excursion boxes by a larger excursion box can be repeated until, at the top of the

hierarchy, a single n -dimensional excursion box encloses all entities contained in the n -dimensional space. Such a hierarchical organisation is well suited to engineering systems which usually consist of multiple levels of assemblies of sub-assemblies of piece parts. For example a motor vehicle is an assembly which contains the engine. This is itself an assembly containing, amongst other things, the valve block, which in turn contains valve assemblies which finally consist of individual piece parts like the valve itself. As this example shows, the hierarchical organisation converts the problem of identifying a particular part from the problem of finding the part directly in one selection from a very large set, to one where the set of possibilities is gradually narrowed down through many selections from very much smaller sets. Of the two methods, the latter tends to be more rapid in yielding the answer.

Thus excursion boxes may be used as a qualitative description of the region of space occupied by some entity by providing a "simple" (orthogonally decomposable) bounding envelope representing spatial uncertainty. This is similar to the qualitative representation of a (one-dimensional) continuously varying parameter by means of a bounding interval.

The excursion box which approximates a given entity and can thus be interpreted as both a qualitative description and as a representation of spatial uncertainty. In the representation of spatial uncertainty, excursion boxes provide a single formalism for both the issue of uncertainty of dimension and uncertainty about the complexity or nature of the shape. In both cases, replacing the single large excursion box by its content of smaller excursion boxes will provide a more accurate description of both its dimensions and emerging shape.

Indeed this process may be continued beyond the base of the hierarchy where each excursion box just contains a single convex polytope, by, for example, systematically decomposing that excursion box into smaller excursion boxes, using these to divide up the original polytope and finally finding the excursion boxes of each of the resulting smaller polytopes. However, this will eventually result in two kinds of excursion boxes (those which straddle the boundary, and those

which only contain interior points of the polytope and thus need not be considered any further) and therefore increase the complexity of the algorithms. Nevertheless, the adoption of this approach would lead to a system which operates exclusively with excursion boxes which are decomposed as often as required. In this context the excursion box should be considered as a region of spatial uncertainty which has associated with it the means of decreasing this uncertainty should this be required : in the case of an excursion box which contains many entities, the reduction of uncertainty is achieved by switching attention directly to the excursion boxes of the contained entities, and in the case of an excursion box containing only a single convex polytope simply apply the method suggested above.

The principles underlying the idea of excursion boxes as outlined above are not entirely new as is illustrated in the next Section.

3.3.3 Related Concepts

The ideas of hierarchical organisation of increasingly more accurate approximations has been applied to geometric problems before. This occurred both in the area of graphical representation of two-dimensional curves in the form of Strip Trees [BALL81] and in the area of Artificial Intelligence with the Mercator representation of (two-dimensional) spatial knowledge [DAVI83].

The Strip Tree representation [BALL81] consists of a binary trees with additional information stored at each node. This information describes a rectangular strip which fully contains part of a two-dimensional curve and pointers to a pair of further strips at the next lower level in the Strip Tree which together form a better approximation to the curve than the initial strip. The lower levels in this tree thus correspond to finer resolutions, that is smaller and narrower strips. The Strip Tree representation is closed under intersection and union operations which may be carried out at any resolution. All operations such as intersection or point membership tests will prefer to operate at the lowest possible resolutions which results in savings of computational time.

The Mercator system [DAVI83] is concerned with the representation of shapes and positions in two-dimensional geographical map. The context is that of a (hypothetical) mobile robot reasoning about geographic real world knowledge which may be both incomplete and inexact. The basic element of the representation scheme are straight lines which are used to describe (polygonal) object boundaries. As many objects do not have straight line boundaries, the representation scheme is only approximate. Multiple approximations are stored in a hierarchy organised in terms of “grain-size” which quantifies the degree of the approximation. Relative positions and dimensions are represented imprecisely by means of “fuzziness” intervals. The difference between “grain-size” and “fuzziness” is that “fuzziness” bounds the uncertainty on dimensions, whereas “grain-size” bounds the uncertainty in the nature of a shape.

Finally, the concept of excursion boxes must be related to that of representing objects through Octree-like solid modelling systems (see Section 2.2). The key difference is that Octrees consists of an enumeration of spatial occupancy based on the decomposition of *space* into regularly shaped subspaces, whereas the excursion boxes are positioned relative to the *object*. Furthermore plain Octrees only involves decomposition cells which, at the leaves of the Octree, only store one piece of information, namely whether they are full or empty. Polytrees [CARL85], a generalisation of Octrees, allow in addition slightly more complex leaf cells such cells containing either a vertex, an edge or a surface. Finally, excursion boxes at the lowest level of their hierarchy contain still more complex information in the form of a whole convex polytope.

3.3.4 Excursion Box Summary

The concept of excursion boxes, although it does not add directly to the representation of continuously varying parameters, is nevertheless valuable as it

1. appears as a natural generalisation of the qualitative representation system to higher-dimensional information.
2. has acceptable storage requirements.
3. permits any information to be structured into simple box shaped regions of space which can be interpreted as the best current upper boundary on the (spatial) uncertainty of shape and dimension.
4. means that information need only be accessed up to the level of detail that is needed for the current problem.

The next Section will in fact pursue this idea of representations of complex shapes through use of gradually more precise approximations as and when needed.

3.4 CONSTRUCTIVE DEFINITIONS

The previous Section investigated the concept of excursion boxes. These could be interpreted as an extension to higher dimensions of qualitative representations schemes which has the additional property of being hierarchical. It was further suggested that an excursion box in the hierarchy can be regarded as a self contained unit which comprises all the information needed to obtain a more detailed set of excursion boxes should the initial box not provide a sufficiently detailed approximation. The decomposition information takes the form of either a list of contained excursion boxes or a *constructive definition*, that is, an algorithm for dividing a convex polytope into smaller pieces. The idea underlying this latter approach, its possibilities, origins and foundations are investigated in this Section.

3.4.1 Origins of Constructive Definition

Before the introduction of computer based drawing aids, all the engineering drafting work had to be performed manually on a drawing board with but a few tools such as straight edge, set squares and compass. These tools are sufficient to deal with most drafting needs directly. However there are situations where more complex, often non planar, curved shapes resulting from intersections or developments must be represented. As these complex shapes cannot be drawn directly with the available tools, graphical methods of construction are used to obtain individual points on these curves. The greater the number of constructed points, the more accurate the approximation to the complex shape becomes. For a complete description of the complex shape all, infinitely many of, its points would have to be constructed.

These methods of construction form the engineering origin of the concept of constructive definitions and will now be considered in more detail.

The method of construction begins with the enumeration, within a certain level of detail, of (often equidistant) points on a simple geometric figure such as a line segment or a circle. The number of points chosen at this stage corresponds to the

number of points to be constructed on the complex shape. In the following, main stage, the same method of construction is applied to each of these points. These constructions invariably consist of drawing so called “projection lines” which are intersected with identified existing entities to yield new points. This process is repeated with the new set of points in a predefined sequence dictated by the particular construction problem until points which lie on the complex shape have been obtained. The projection “lines” are restricted to those which can be easily drawn with the available tools, that is either straight lines, or circles. They are specified in the different views of a traditional drawing as being parallel to existing lines (often centre-lines, axes of symmetry or coordinate system axes) or about an existing, identified centre. In the final stages of a constructive definition of a complex shape, these projection lines are often used to link corresponding points in the different engineering drawing views. Specific examples of these methods of construction can be found most textbooks on technical drawing [THOM80, ABBO78].

The key idea is that complex geometric entities can be fully specified by a finite algorithm consisting only of primitive operations applied to a finite number of already existing entities. This is the main idea which underlies the proposed constructive definitions.

3.4.2 Constructive Definitions

However, the representation of a (complex) geometric shape in terms of some of the points contained within it is not sufficient for the integration with the proposed geometric knowledge representation scheme. What is needed is the addition of a (polyhedral) envelope which completely encloses the shape between the constructed points and which may be used as an approximation to it. This would involve constructing not only individual points on the complex shape, but also simple bounding polytopes of what occurs between these points. This polytope would in turn be approximated by an excursion box, whose reduction into smaller excursion boxes would consist of this constructive definition, that is, the method of construction of (further) points within the envelope of the approximating polytope, together with the construction of the set of resulting, smaller polytopes of approximation, whose excursion boxes would form the required more accurate excursion box based description. (Each of these resulting smaller excursion boxes could then, if necessary, be itself further decomposed by the same constructive definition without needed to refer to any "global" pieces of information).

This then forms the definition of the "constructive definition" of a piece of geometric knowledge. The the concept of "lazy" or "demand driven" evaluation as implemented within some functional programming languages [HUGH85] is clearly related to the proposed use of constructive definitions which can now be recognised as specifying an infinite data structure and therefore as a means of representing non-polyhedral entities.

Note that in the manual drawing board constructions, the bounding polyhedra were not required and therefore not constructed. Instead the detail of the complex shape between constructed points is simply added in by what amounts to an arbitrary free hand sketch.

However, the proposed approach does requires additional operators. Apart from those that are needed to link a constructive definition with a polytope (or alter it), the main new operator is the one which will yield a certain number of points

(preferably evenly distributed) within its operand entity. For entities which are constructively defined this process simply consists of applying their constructive definition, but will not necessarily yield evenly distributed points. This is effectively a recursive definition which needs a base case in order to become practical.

The base case occurs when the points in a non constructively defined polytope need to be enumerated. Here again the situation can be simplified by considering the lower dimensional cases and using a recursive definition to deal with the higher dimensional polytopes.

In the (trivial) zero-dimensional case, the enumeration of a single point only yields itself. In one-dimension, the points on an interval or line segment must be enumerated. This can be achieved easily in a system using coordinate geometry by bisecting the line segment to yield its mid point as well as two smaller line segments. The process of bisection can then be repeated as often as necessary on the resulting smaller line segments to obtain more points. This, if necessary, repeated subdivision operation on an interval is in fact the only new primitive operation that is required.

Points in a higher dimensional polytope can be enumerated in a systematic manner by means the excursion box. The interval subdivision operation is applied to each of the excursion intervals whose Cartesian product formed the excursion box of the original polytope. The Cartesian product of the subdivided excursion intervals will yield a set of excursion boxes whose union is equal to the original excursion box. This set of smaller boxes can be intersected with the original polytope to yield a set of smaller polytopes, at most one in each of the smaller excursion boxes. This results from the fact that excursion boxes are themselves polytopes and intersection operation on polytopes is closed. Each of these smaller polytopes is defined by means of its vertices, some of which will be the vertices of the original polytope, while others are new. These 'new' vertices form the desired enumerating points and lie on the boundary or in the interior of the initial polytope. Effectively the enumeration of points in higher dimensional polytopes is itself defined constructively.

The introduction of constructively defined entities does raise the issue of how two constructively defined entities might be compared, in particular as different methods of construction can be used to specify the same piece of geometric knowledge. Two different options can be pursued in overcoming this problem.

The first method remains entirely within the excursion boxes approach. It simply consists of obtaining and comparing ever more precise excursion box based approximations of the two shapes. If the two entities being compared are different then this method will find that answer in finite time. If however they are equal, then this method will never terminate, or, in other terms, will only be able to give an approximate answer to within the current size of the excursion boxes. This may nevertheless be acceptable for engineering problems where numerical values and geometric shapes are often only known to lie within some tolerance bands, which must, however, be equal to or greater than the size of excursion box used for the approximate comparison answer.

The second alternative is that of using a symbolic method to determine equality. This would involve two stages, the first to convert the constructive definition into “sentences” in an appropriate language applying a decision method to determine whether both sets of sentences are included in each other. The “appropriate” language might well be that of elementary real number algebra or a slight modification of it. The theoretical issue of the existence of such a mechanical decision method for elementary algebra and geometry in n -dimensional Euclidean space has already been solved. Tarski [TARS51] describes such a method which will always compute an answer in a finite number of steps, but does not provide an estimate of the amount of resources needed by the algorithm to do so.

This area of constructive definitions for the representation of geometric knowledge therefore clearly requires further research. This applies in particular to entities obtained from using constructive definitions in conjunction with the join (or convex hull) operator.

The next Section will now explore the different interpretations that can be placed on geometric knowledge.

3.5 PLURALITY OF INTERPRETATION

The possibility of placing more than one interpretation on a given piece of geometric knowledge has already been encountered. For example, a Solid Modelling system always interprets geometric models as representing the shape of some physical object and therefore only permits them to be manipulated as such. An alternative interpretation of a geometric representation is as the graph of a relationship or function. This is the interpretation which prevails in qualitative reasoning systems. This Section will therefore investigate how many distinct interpretations can be placed on any piece of geometric knowledge and how these may be used.

First consider the interpretation of a geometric representation in n -dimensional space as the graph of a relationship or function.

This perspective requires the n -dimensional space to be partitioned into two parts, namely a d -dimensional subspace equal to, or containing the domain of the function or relationship and the $(n - d)$ -dimensional orthogonal space containing its range.

The process of applying the function or relationship to a point in the domain space can in fact already be constructed using the primitive operations of the proposed system for the representation of geometric knowledge. The starting position is a point in the d -dimensional domain space. The Cartesian product of this point with the $(n - d)$ -dimensional range space is added to the n -dimensional space containing the geometric representation of the graph of the function or relationship. The intersection of these two geometric representations can then be calculated. If the initial point was outside the domain of the function or relationship, then the outcome will be the empty set and there is no image of the point under that particular function. Otherwise, the intersection will yield some geometric entity that can be projected (orthogonally) onto the $(n - d)$ -dimensional range space to yield the geometric representation of the image of the initial point. If the graph is indeed the graph of a function, then the image is also a point,

otherwise it may be a more complex entity such as, for example, an interval (or more precisely its geometric equivalent, a line segment).

Note that if the graph of a function or relationship is known, its inverse can be computed just as easily simply by inverting the roles of the range and domain spaces. Indeed for the graph of a function or relationship in an n -dimensional space, the full range of possibilities consists of all ways of partitioning the n -dimensions into two sets, one for the domain space and one the range space.

In enumerating all possibilities, this approach also introduces the limiting cases where either the range space or the domain space is empty while the other consists of the whole n -dimensional space. In the first case the function or relationship has no range (that is the range is empty) and in the second case it has no arguments (that is the domain is empty). Even though neither case has any obvious interpretations, the following possibility nevertheless exists :

A 'function' with no range is not a usual function in that applying it, in turn, to each point of its domain yields nothing. In fact doing so, amounts to enumerating the points in the domain. As in this case the domain is in fact equal to the graph of the 'function' or relationship this piece of geometric knowledge is effectively treated as the set of its constituent points, with the main emphasis resting on the points and not the set as a whole.

The 'function' with no domain yields its whole range from nothingness. As in this case the range is in fact equal to the graph of the 'function' or relationship this piece of geometric knowledge is effectively treated as a whole indivisible set, and may thus be used as a single entity in a way identical to that of a geometric model of a solid object.

If these two interpretations are accepted, then all possible interpretations of a piece of geometric knowledge can be placed under a single theoretic paradigm.

Consider a piece of geometric knowledge in a n -dimensional space, which is partitioned into a d -dimensional sub-space, the 'Domain', and the $(n - d)$ -dimensional

'Range' space orthogonal to it. Then all possible interpretations of that same piece of geometric knowledge lie within one of the following alternative cases :

if	{	$d = \infty n$	the geometric knowledge is considered as an enumeration of its points, which may be used for some construction process. This could be termed an "inward looking" point of view.
		$0 < d < n$	the geometric knowledge is considered as a relationship or function linking some domain in the Domain-space with a range in the Range-space. This corresponds to viewing the geometric knowledge as either relating a whole domain entity and a whole range entity or as relating an enumeration of points in the domain to an enumeration of corresponding points in the range.
		$d = \infty 0$	the geometric knowledge is considered as a whole in its own right, and may be subjected to some set theoretic operation or transformation mapping. This could thus be termed an "outward looking" point of view.

This spectrum of interpretations is clearly broader than that supported by either solid modelling or qualitative reasoning systems. It is even larger than that supported by deduction systems based on Horn Clause logic and that of functional programming.

Horn Clause logic based deduction systems permit an n -ary predicate to be queried with any combination of its n parameters instantiated in order to find the remaining unknown ones. The strict approach to first order logic however prevents the set of all parameter tuples from being treated as a single indivisible whole even though this is theoretically possible [BELL86].

Functional programming [HUGH85] also nearly supports the full spectrum with the first interpretation resembling the principle of demand driven or 'lazy' eval-

uation, the second existing in a restricted form as 'partial evaluation' [MOGE86] while the third is similar to 'program transformations' [DARL81].

The subject of demand driven or 'lazy' evaluation has already been introduced during the discussion of excursion boxes. Partial evaluation is a way of processing a function when some, but not all of its arguments are known so as to obtain a derived function with fewer arguments which are all unknown. This process covers some, but not all of the cases in $(0 < d < n)$ above. In particular, the interpretation as a function means that the value of the parameter which the function computes may be not be fixed, and thus this approach effectively excludes the inverse or partial inverse functions. However, with the convergence of logic and functional languages [DARL85] the full spectrum of interpretations should gradually become available in the future.

Having clarified the spectrum of interpretations which may be placed on a piece of geometric knowledge, the next Section will focus on the qualitative representation of a piece of geometric knowledge.

3.6 QUALITATIVE REPRESENTATIONS

The previous Section described how a same piece of geometric knowledge can be considered as a set of points or as an entity as a whole, or as some kind of function or relationship. This Section will briefly explore the resulting implications for qualitative descriptions of geometric knowledge.

The most important aspects are what kind of qualitative descriptions correspond to each of the different interpretations that may be placed on geometric knowledge and whether these descriptions can be obtained automatically. The relationship with the qualitative reasoning systems described in Section 2.4 is also considered, especially with respect to the kind of additional phenomena introduced through working in n -dimensions. For the purpose of comparisons the representation scheme employed by the QSIM system [KUIP84] will be used in preference to the others in Section 2.4 owing to its generality and absence of implicit assumptions concerning a domain of application.

When a piece of geometric knowledge is considered as a whole or as an enumerated set of points, excursion boxes can be used to qualitatively represent its spatial distribution. In this context the excursion boxes correspond to a generalisation to higher dimensions of the (one-dimensional) intervals used in qualitative reasoning.

In qualitative reasoning the (qualitative) description of a function in fact requires two intervals, one representing the domain (typically an interval of time) and the other the corresponding range, together with an additional piece of information as to whether the function is increasing or decreasing monotonically or is constant (in which latter case the range 'interval' is in fact reduced to a point). Note that the Cartesian product of the domain and range intervals forms the excursion box of the graph of that function.

Thus, when a piece of geometric knowledge is interpreted as a function or relationship, its excursion box on its own is not sufficient as it does not contain any indication of what the function inside the excursion box might look like.

Therefore the excursion box based representation must be augmented by some additional pieces of information.

Although the vocabulary of the Qualitative Reasoning (and that of the QSIM system [KUIP84], in particular) appears to be a good starting position it nevertheless needs to be extended to take into account the additional phenomena occurring in higher dimensions.

Functions of more than one variable (whose graphs would be thus have to be represented in three or higher dimensional spaces) could be represented qualitatively by the Cartesian product of the qualitative descriptions as perceived along each coordinate axis. That is, for a function of n variables, its domain would be decomposed into excursion boxes. Each excursion box would be such that the function were monotonically increasing, decreasing or constant (over the corresponding range) with respect to one of its variables independently of how the others were chosen within the part of its domain enclosed by the excursion box. That this system of description is sufficient for representing any continuously varying function (in higher dimensions) clearly follows from the fact that the vocabulary of qualitative reasoning is sufficient for functions of one variable.

Note that one higher dimensional phenomenon which has to be taken into account is that in a qualitative description of a function of n variables the excursion box of the domain will also, in general, include points which are either not in the domain of the function, or are in the domain of the function, but not in the part to which the current qualitative description applies. The excursion box of the domain will thus have to include the (possibly constructive) definition of the actual part of the domain that is concerned.

The approach described so far clearly only applies to single valued functions of several variables. However, there are many pieces of geometric knowledge which may be considered as mappings but which are not single valued and therefore are not 'functions' according the mathematical definition. These can be further distinguished into those mappings which, for a point in the domain, yield a discrete set of corresponding image points and those that yield a line segment,

that is an interval of values. The former can be considered as a finite collection of 'proper' functions (and the qualitative descriptions obtained accordingly) while the latter must be represented by a pair of 'proper' functions, one describing the upper bound and the other the lower bound of the solution interval(s).

Again the approach described so far only applies to functions which yield a one-dimensional image for each antecedent in the domain. However, mappings which yield an r -dimensional image may be treated as a set of r functions which each yield a one-dimensional image, and thus be represented by r qualitative descriptions obtained accordingly, whereas the r -dimensional image as a whole would be represented by its excursion box.

This completes the outline of the qualitative representations corresponding to all alternative interpretation cases of items of geometric knowledge. Since there are multiple ways of interpreting (and using) a piece of geometric knowledge the next question that therefore arises is can the qualitative representations be derived or converted automatically ?

The fact that the proposed generalised solid modelling system underlying the representation of geometric knowledge uses convex hulls as its primitive representation unit will be of assistance in deriving automatically the qualitative description from a given geometric model. Obtaining the excursion box for a given geometric model does not present a problem (see Section 3.3). The above investigation further showed that only one basic case need be considered, namely that of a n -dimensional geometric model being interpreted as the graph of a mapping from an $(n - 1)$ -dimensional domain into a one-dimensional range. The basic unit of the geometric model will be a n -dimensional polytope, which is a compact convex set and therefore homeomorphic (that is in a one to one correspondence which is continuous in both directions) to the unit n -dimensional ball [STOE70 p124]. The boundary of the polytope can therefore be decomposed into two functions corresponding to the lower and upper 'hemispheres' bounding the interval of image values corresponding to each point in the $(n - 1)$ -dimensional domain. The (faceted) boundary of the polytope can be subdivided into regions of equal qualitative description based on the orientation of the surface facet normals.

A qualitative description of a constructively defined geometric model can either be obtained from one of its approximations or possibly from the constructive definition. However, this area requires further specific research which is beyond the scope of this thesis.

Another issue which would also require further consideration is the detailed definition of the qualitative vocabulary, in particular how it can be made hierarchical so as to provide even more concise (but vaguer) descriptions in parallel with the excursion box hierarchy. Considering the simpler case of a qualitative description of a function of a single parameter, an even vaguer qualitative description could only be obtained by augmenting the qualitative vocabulary by such concepts as it non-monotonic but generally increasing, it decreasing or it constant/indeterminate to qualify larger sections (excursion boxes) of the functions behaviour. This amounts to a kind of filtering operation in which small scale variation is 'ignored' in favour of the global trend, similar in nature to the 'human' description of a line, drawn free hand, as 'straight' rather than irregular and undulating.

Finally, the purpose of qualitative representations is to provide the language for a system of representations which can be tailored dynamically, through demand driven evaluation and qualitative reasoning to the level of detail required for the resolving of the current problem.

In particular, the outcome of one of the qualitative reasoning systems (QSIM, see Section 2.4) is to construct an additional qualitative representation of some continuously varying parameters of time using the supplied sets of qualitative equations. The computed descriptions are in addition to the (default) minimal qualitative representation in terms of the interval $(-\infty, +\infty)$. The purpose of qualitative reasoning about geometric knowledge would similarly be the construction of additional qualitative representations in addition to those arising exclusively from the different interpretations that may be placed on a piece of geometric knowledge within its space.

In a design context, qualitative reasoning could, for example, be used to verify whether a proposed, more detailed (geometric) description does indeed correspond to a vaguer, qualitative description which was established earlier in the design process and thus confirm that the design is indeed converging onto a solution.

In particular, it should be recognised that the ability to change between the different interpretations that may be placed on a piece of geometric knowledge is important as it occurs (often implicitly) in reasoning about geometry in a design context.

The most explicit example is that of a linear cam whose working profile directly matches the graph of the function linking the cam follower displacement to the position of the cam. In one case the interpretation is as a geometric shape and in the other as a function.

Other examples include deducing the way the shape of a cross section changes as the cutting plane is swept across the model of an engineering component. Here the purely geometric shape data must be considered as functions which cause the edges of the cross sectional area to change in a certain manner. In particular the boundaries of the geometric shape are considered as representing a 'function' from a one-dimensional variable, namely the position of the cross section along the solid model, into the two-dimensional plane of the cross section. Such reasoning might be used to locate regions of interest such as for example minimum or maximum cross sections in the given component.

This completes the brief investigation into qualitative representations of geometric knowledge. The next Section will recapitulate the ideas presented so far so as to obtain a balanced overview.

3.7 GEOMETRIC KNOWLEDGE SUMMARY

Starting from the concept of *geometric knowledge* as any piece of information that is best described by a continuously varying parameter in some n -dimensional space, this chapter has outlined a representation system which respects its particular properties.

A piece of geometric knowledge is represented by the set of points over which a continuously varying parameter may range. As in the general case, this set of points has an infinite number of elements, difficulties arise in selecting a finite description of it. In fact there are indefinitely many ways of providing such a descriptive taxonomy. Furthermore, providing such a finite descriptive taxonomy in isolation from the problem to be solved is not desirable in a general design context for the reasons outlined in chapter 2. Therefore a mechanism for providing a finite description which matches the current requirements is needed. In the context of a continuously varying parameter the finite description is achieved by dividing its set of (in general, infinitely many) points into a finite number of equivalence classes.

If the continuously varying parameter is one-dimensional then the equivalence classes partition it into a finite set of intervals. The *meaning* of the intervals is given by describing how they relate to a corresponding equivalence class of another continuously varying parameter. This is done by describing how both these pairs of intervals correspond and whether their points enumerate in the same order (that is, the mapping between them is monotonically increasing), in the opposite order (that is, the mapping between them is monotonically decreasing) or is undetermined and irrelevant (that is, the mapping between them is constant).

This novel interpretation on the representation scheme of qualitative reasoning is then generalised to continuously varying parameters of higher dimensions. In higher dimensions the one-dimensional intervals of qualitative reasoning are generalised into excursion boxes as a means of qualitatively describing an n -dimensional continuously varying parameter.

However, the extension to higher dimensions also introduces additional phenomena. The most important one is that, unlike its one-dimensional counterpart, a continuously varying parameter of a higher dimension will *not* necessarily range over all values of its containing excursion box. In the one-dimensional case, the excursion interval, defined as the smallest enclosing interval of the continuously varying parameter, will *only* contain values which the continuously varying parameter actually adopts. The implication for higher dimensions is that an additional mechanism must be provided to specify the actual (shape of the) set of points over which the continuously varying parameter ranges.

The proposed method of specifying this shape consists of a generalisation to higher dimensions of a polyhedral solid modelling system. The theoretical basis for the generalisation of the concept of a 'solid' to higher dimensions is provided by the 'join' (or convex hull) operator from a modern (non-dimensional) treatment of convex set theory.

The *meaning* of the excursion boxes, is simply obtained by using the method of qualitative physics separately along each axis and using the combination as the *meaning* of the excursion box. Effectively the *meaning* of the excursion box is the Cartesian product of that of its projected excursion intervals along each axis. Note that this imposes certain constraints on the content of an excursion box when used for this purpose.

This basic representation system for geometric knowledge is then augmented by additional ideas.

The first one is the idea of a hierarchy within the representation system. In an engineering context this idea is a natural one which corresponds well both with the repeated description of systems in terms of sub-systems and with the recommendations from methodical design where a function (in the sense of 'purpose') structure should be decomposed into (alternative) sub-functions.

The purpose of the hierarchy is to obtain a gain in efficiency through operating at those levels which only provide just sufficient resolution for the current purpose and thus avoid wasted resources arising out of an increased search space caused

by an unnecessary level of detail. Therefore the level of required detail should effectively be driven by the demand rather than being imposed at the outset.

Starting from the base level of qualitative descriptions outlined above, the hierarchy may be extended 'downwards' to provide a more detailed and specific representation and 'upwards' to provide a less refined but more global view. Downward extension does not provide any theoretical difficulties as the qualitative nature of the description remains the same while the number of ever smaller excursion boxes increases. Upward extension does, however, require additions to the qualitative vocabulary to represent more approximate descriptions which carry less information as the corresponding excursion boxes grow in size.

The concept of demand driven (or 'lazy') evaluation of the hierarchy can also be used to represent increasingly more accurate, finitely faceted approximations to smoothly curved (or infinitely faceted) shapes and thus effectively extend the representational ability of the proposed scheme without a corresponding increase in complexity of the modelling algorithms.

The second idea is that of flexibility. A piece of geometric knowledge represented by a continuously varying parameter in an n -dimensional space may be interpreted in different ways. The set of all possible interpretations was established theoretically as a kind of 'mapping' from some of the n dimensions into the remainder. The representation scheme supports all of these and preferably should permit changing between them.

The third and final idea is the concept of infinity, or more precisely points at infinity. This is required by the qualitative reasoning system for several reasons. The qualitative representation of a number (that is a continuously varying parameter in one dimension) is represented by the interval $(-\infty, +\infty)$ which expresses the fact that it is a real number by describing the set of *all* possible values that it might adopt. The concept of points (or in this case, values) at infinity is thus required to represent universal quantification. An alternative use results from a slightly different interpretation in which the description $(-\infty, +\infty)$ is regarded as *restricting* the set of possible values of the number. Thus a set of

additional constraints on the number could, through the qualitative deduction process reduce this to a smaller interval. Conversely, a set of additional constraints which a deduction process reduces to the interval $(-\infty, +\infty)$ can be identified as adding nothing new, that is the concept of infinity can also be used to express absence of information. Thus infinity can be used to express both the concepts of 'everything' and 'nothing' within a same representation scheme. The theoretic foundation of augmenting the proposed modelling system in a consistent manner by the addition of points at infinity follows from results obtained in general topology.

Overall the proposed system for the representation of geometric knowledge consists of many individual ideas which when united into a single system form a very coherent and novel whole. In particular, it proposes a unified theoretical view of the set of all interpretations that may be placed on a geometric model, while leading further along the path started by qualitative reasoning to a new method of representation in which excursion boxes are the key representational element.

The new method of representation that is being considered can be best described using an analogy from control theory. Instead of the knowledge representation being defined exclusively by the builder of the knowledge based system (as in an open loop control system where the control input is directly defined by the user), the knowledge representation is constructed dynamically from the user input and the current state of the problem through the combination of demand driven evaluation and qualitative reasoning. This is reminiscent of a control system with a closed feedback loop where the control input is obtained from a combination of the user input and the current state of the system being controlled. By analogy the representation system being aimed at might best be called a 'closed loop' representation system where the control signals are carried in excursion boxes.

The (longer term) objective of this work is to provide the designer with far more flexibility to explore the consequences of design decisions by providing a uniform representation of a large category of engineering information, whose special properties have not before been investigated or acknowledged.

The key limitation is that, in spite of their complexity, pieces of geometric knowledge are but the primitives of the kind of information that must be represented for practical design purposes. In fact geometric knowledge plays a similar role to that of an 'Horn Clause' in a knowledge based system whose inference engine is a deduction system based on Horn Clause logic. Therefore a further mechanism must be provided in order to structure pieces of geometric knowledge into larger units so as to approximate the kind of 'real world' knowledge that is required for the purposes of design.

For example a solid object (in general) remains the same whether it is in one location or another. This property of solid objects means that a piece of geometric knowledge representing an engineering component will in fact be an equivalence class over all rigid motion transformations (which have six degrees of freedom and can therefore be represented in a six-dimensional space) that it can be subjected to. Qualitative Process Theory [DEKL84] (see Section 2.4) proposes a method of representing processes of change.

Furthermore, like other design support systems [BATA83] the effectiveness of the proposed approach will rely heavily on the interface presented to the user, an aspect which is not considered in the framework of this research.

This completes the more theoretical aspects of this research and the following Chapter will consider issues surrounding the implementation of a system for the representation of geometric knowledge.

Chapter 4. IMPLEMENTATION ISSUES

4.1 OVERVIEW

This Chapter will consider issues surrounding the implementation of the ideas presented in Chapter 3.

The sequence of implementation of a system of the scale and relative complexity of the one outlined in Chapter 2 must to some extent reflect the interdependence of its constituent ideas. The generalisation to n -dimensions of a polyhedral solid modeller based on the Join (or convex hull) operator is a relatively independent element of the proposed system in the sense of not being heavily dependent on other parts while the centre of the representation system, the excursion box hierarchy, is not really feasible without it. This generalised modeller is, however, also one of the most complex parts of the proposed system.

Therefore, a possible implementation path begins with the generalised modelling system, augments it by the excursion box hierarchy, then introduces constructive definitions and qualitative descriptions and finally considers the generalisation of the qualitative deduction system.

However, due to the amount of software that would be required an implementation of a complete system is far beyond the scope of this research. The actual implementation was therefore restricted to illustrating the principle operators for the generalised, n -dimensional modelling system. As a consequence the more detailed implementation issues will concentrate on the generalised polyhedral modeller. These are divided into two parts, the data structures and the algorithms; they are discussed in the following Sections and are followed by a brief description of the current state of implementation of a prototype.

4.2 DATA STRUCTURES

The main data structures of the generalised modelling system are associated with the representation of points, of convex polytopes and of lists of these. Each of the data structures must be augmented by its excursion box and qualitative description. These issues are discussed in turn in the following Sections.

4.2.1 Representation Of Points

The representation of points is in usual circumstances not a subject worthy of long discussion. However, because the proposed modeller is generalised to permit the representation of points at infinity in an n -dimensional space new problems arise.

In a system based on coordinate geometry a point P is usually represented as a tuple of real numbers :

$$P = (x_1, \dots, x_n)$$

Each of the x_i (where $1 \leq i \leq n$) represents the distance from the origin of the coordinate system to the projection of that point onto the i th coordinate system axis.

When the point coordinate values are restricted to real numbers, only finite points can be represented. To represent points at infinity, the representation of the coordinate values must be augmented to represent the two additional values of $-\infty$ and $+\infty$. As the usual representation of floating point numbers in a computer system does not include infinite values the burden of representing these falls onto the programmer. One of the simplest methods of doing so would be to store an additional tuple of values for each point :

$$P = (x_1, \dots, x_n), (a_1, \dots, a_n)$$

The value of each a_i value (where $1 \leq i \leq n$) indicates whether the corresponding coordinate value x_i is finite or infinite. For example, if the a_i are binary variables then a value of '0' indicates whether the x_i is finite and a '1' means that x_i coordinate is at infinity, $+\infty$ if $x_i \geq 0$ and $-\infty$ otherwise.

Another method of representing points at infinity is through the use of homogeneous coordinates. Here a point P in n -dimensional space is represented through $(n + 1)$ coordinates $(x_{h1}, \dots, x_{hn}, \lambda)$ which are related to the original n coordinates $(x_1, \dots, x_n)$ by

$$x_i = x_{hi}/\lambda \quad \text{for } 1 \leq i \leq n$$

When λ is non-zero the corresponding point is finite, when λ is zero and at least one of the x_i is non-zero then the point is at infinity — effectively the n coordinates $(x_{h1}, \dots, x_{hn})$ represent a vector rooted at the origin and pointing towards it. When all the $(n + 1)$ homogeneous coordinates are zero then the point is indeterminate — a condition which should be avoided.

Neither of the two representation systems is ideal for use in the proposed generalised modelling system. The reasons can best be explained by means of the simple task of representing a whole straight line. A straight line is a convex set of points with an extreme point (or vertex) at each (infinite) ‘end’. Therefore it can be represented as the join (convex hull) of these two ‘end points’.

With homogeneous coordinates, only lines passing through the origin of the coordinate system can be represented in this manner. Other lines simply cannot be represented as there is no way of specifying the end points. Therefore, lines not passing through the origin must be represented as a list of two opposing rays with same (finite) starting point. Furthermore, similar strategies must be employed in the representation of infinite excursion boxes which is inconvenient as it increases the complexity of both excursion box and Cartesian product related algorithms. However, the advantage of homogeneous coordinates is that they are already well understood and used in existing systems to simplify certain geometric transformations [FOLE82].

The first representation system proposed above, does indeed overcome this problem, but only at the expense of introducing another. The reason for which the homogeneous coordinate representation failed to provide an elegant means of representing the extreme points of lines not passing through the origin is that it cannot express a finite separation occurring over an infinite distance, that is

it cannot represent two points at infinity which are only separated by a finite amount. The first representation system proposed above can clearly represent such distinctions, and in so doing, effectively permits the representation of infinitesimal quantities.

For example, consider a (two-dimensional) plane with a Cartesian coordinate system whose first coordinate axis is called x and the second y . The x -axis can be represented as the join (convex hull) of the points $(+\infty, 0)$ and $(-\infty, 0)$. Now consider the point at $(+\infty, 1)$. It is clearly a point at infinity but is also separated from the positive end of the x -axis by only a unit step along the y -axis. If this point is "joined" to the $(-\infty, 0)$ point the result is a straight line which is finitely separated from the x -axis by a unit step along the y -axis at one 'end' and intersects the x -axis at the other. The slope of this line is $\frac{\Delta y}{\Delta x}$, where Δx is $(+\infty - (-\infty))$ and Δy is unity. This yields a slope which is only infinitesimally (that is not finitely) different from zero and thus, to a finite observer, appears to be parallel to the x -axis, separated from it by an amount depending on the relative magnitudes of $+\infty$ and $-\infty$. If these relative magnitudes are equal the separation will be a distance of half a unit. In this latter case, the line is in effect indistinguishable (by a finite observer) from the line whose extreme points really are $(-\infty, 0.5)$ and $(+\infty, 0.5)$. To avoid problems (such as multiple orders of infinitesimal and infinite values) resulting from such infinitesimally "warped" entities, additional algorithms would have to be provided to replace these by their non finitely distinguishable but well behaved counterparts. This additional complexity surfaces in particular in the implementation of the proposed join (or convex hull) operator.

A further, more serious problem arises when attempting to represent the end points of lines whose slope is finite but non zero. All 'end' points of such lines which lie in the same quadrant of a plane (or octant of a three-dimensional space, etc) cannot be distinguished within this representation scheme. This scheme is therefore only really suited to the representation of excursion boxes which are bounded by entities which are parallel to coordinate system entities.

The purpose of introducing the concept of infinity was to act as an 'conceptual' imagined boundary to effectively unbounded entities. It is therefore similar to the concept of infinity as used in limit calculations in mathematical analysis. Even though the concept of an infinitesimal quantity has a long history of causing problems in mathematics, the area of non-standard analysis has proved that it is well founded. Non-standard analysis has further proved that although the concept of an infinitesimal quantity does not admit a unique definition, it can nevertheless be used to obtain non-standard proofs of classic results in analysis [DAVI77]. However, for most engineering applications infinitesimal quantities are not required and therefore will not be included within the proposed representation scheme for geometric knowledge.

Returning to the issue of the representation of points, the fact that spaces of varying dimensionality are supported means that different numbers of coordinate values will, in general, have to be stored for different points. As a fixed sized storage scheme would have to provide sufficient space for the highest dimensional case this is an approach which would be rather wasteful of computer storage capacity. It therefore seems to be more desirable to opt for a dynamic allocation of storage capacity for point coordinate tuples in spite of the additional programming complexity that this entails.

A further consideration that will affect the representation of points is the desire for uniformity in the data structures which in turn would reduce the complexity of algorithms operating on them. In this context a point should be considered as a degenerate case of an excursion box and of a polytope. A point could be considered as an excursion box whose excursion intervals along the coordinate axes each have coincidental upper and lower bounds, or as a polytope which is effectively reduced to the convex hull (join) of a single extreme point, namely the point itself.

These latter considerations, however, require an insight into the representation of polytopes which is provided in the next Section.

4.2.2 Representation Of Polytopes

A polytope is the closed convex hull (join) of its finite set of extreme points (vertices) which may either be finite points or points at infinity.

A representation system for polytopes should reflect their particular properties. These are firstly the fact that a polytope is completely determined by its set of extreme points (vertices) and secondly that the boundary of a polytope consists itself of polytopes of next lower dimensionality. For example, the boundary of a polyhedron, a three-dimensional polytope, consists of two-dimensional faces, each of which is a polygon, that is a two-dimensional polytope.

A useful characterisation is provided by the mathematical interpretation of the term *component* of a polytope. A *component* of a polytope P is defined as any polytope obtained from the join (convex hull) of a subset of the vertices of P . The set of all *components* of a polytope P comprises each of the extreme points (vertices), one-dimensional edges, two-dimensional faces, ..., $(n - 1)$ -dimensional (hyper) faces and finally the n -dimensional polytope P itself.

The description of a polytope simply as a list of its *components* accessed by dimension has the drawback of not being a self similar representation scheme; that is, the representation of a face within a larger polytope is different from that face represented as a polytope in its own right. The reason is that in the former case the *components* of the face are intermingled with those of other faces of the larger 'parent' polytope whereas in the latter case they are not. However, the ease of access to *components* of a given dimension offered by this scheme does make it a valuable addition to the self-similar representation schemes outlined below.

The advantage of a self similar representation system is that it enables the use of recursive algorithms and thus reduces the complexity of implementing systems capable of operating in n -dimensional spaces.

Two representation systems which have this required property of self similarity are based either on the boundaries or the extreme points. Again neither approach is ideal.

In the representation system based on boundaries, each n -dimensional polytope is represented by a list of (pointers to) each of its bounding $(n - 1)$ -dimensional faces. In the case of a zero-dimensional polytope (that is a single point), this list is empty.

The advantage is that access to the $(n - 1)$ -dimensional boundary components of a given n -dimensional polytope is very easy and efficient, a feature whose importance follows from this operation's being frequently required by the geometric modelling algorithms. Access to the extreme points (vertices) of a polytope or one of its components is however far more complex and inefficient as it involves traversing the whole boundary component structure of the polytope.

In the representation system based on extreme points a polytope essentially becomes similar to a hyper-graph. Whereas an ordinary graph is defined as a set of vertices and a set of edges, (where each edge is in turn represented by the pair of vertices that are joined by it) a hyper-graph presents an altogether more complex structure. This results from the edges being generalised from joining two points to hyper-edges capable of joining many points. In a polytope these hyper-edges correspond to the *components* of the polytope which, as each vertex belongs to a space with coordinate geometry, can be partitioned by dimension. Thus a polytope and all of its components, can be represented by a subset (list) of all vertices (points) in the space.

The advantage of this representation scheme is that it clearly provides very efficient access to the extreme points of any polytope. However, access to the boundary components of a polytope is more complex as these must first be determined. This is inconvenient with respect to some of the modelling algorithms where some operation must be performed in turn on each of the boundary facets of a given polytope.

Combining all the above representation scheme would yield one which would be very efficient from the point of view of performing modelling operations, but would be rather extravagant in its use of computer memory and difficult to maintain owing to the incurred redundancy of the information. The selection of a representation scheme for the polytopes thus reduces to a well known compromise between the computational resources of speed and storage requirements.

Again, as lists of varying length have to be stored, dynamic data structures must be used. This raises a further problem which arises in all the above systems when the representation of a polytope must be transferred between the computer memory and a permanent storage medium. The meaning of a list of pointers (either to the boundary facets or to the extreme points of a given polytope) is totally dependent on the actual position in the computer memory of the polytope data structures. These lists of pointers must therefore be changed by some appropriate additional algorithms in favour of some implicit or explicit labelling system as the pointers lose their meaning when they are moved out of the computer memory.

The final Section now considers the representation of excursion boxes.

4.2.3 Excursion Boxes

As an excursion box is defined as the Cartesian product of the excursion intervals of some geometric entity along each of the coordinate axes of the space, it can fully described by 2^n numerical values, where n is the dimension of the containing space.

As excursion boxes are the fundamental information storage vehicle of the proposed representation system, they should be regarded as the backbone to which all other information is attached. This additional information is divided into two categories, that relating to the hierarchy of excursion boxes and the qualitative description of the contents of the box.

In order to support the demand driven access to the lower, more detailed levels of information in the hierarchy, each excursion box must contain, or refer to, a next, more detailed, level of representation. The more detailed level of information can be represented by two lists; a list of geometric entities (polytopes, or finite lists of polytopes) and a list of geometric modelling operations to perform on these. This data structure can in fact accommodate all cases.

'High' level excursion boxes which contain a set of smaller excursion boxes are represented by list of contained entities and a list of operations consisting in the single instruction to enumerate the elements of that list. At a lower level in the hierarchy, excursion boxes which contain only a single polytope will eventually be reached. These are represented by a list only containing that polytope and a list of geometric modelling instructions (as outlined in Section 3.3) for splitting the polytope into 2^n parts (where n is the dimension of the space containing it). In the case of a constructively defined shape, the list of geometric operations associated with the excursion box of a polytope approximating this shape simply consists of the constructive definition. In this case the list of geometric entities must include not only the approximating polytope but also those geometric entities that are needed in the construction process.

When an excursion box containing a single polytope is decomposed into smaller excursion boxes, the list of modelling instructions for the decomposition process

can be re-used without change for further decomposition should this be needed. Therefore the decomposition process only ever needs information which is already available locally from the excursion box data structure.

As already suggested in Section 3.6, the qualitative description of the contents of an excursion box requires augmenting the vocabulary of qualitative reasoning. This necessity originates from three main sources : higher dimensions, flexibility of interpretation and hierarchy.

The effect of higher dimensions is to introduce functions of more than one variable whose qualitative descriptions can be represented as a list of the qualitative descriptions with respect to each variable in turn over suitable subdivisions of the function. Thus, a function of n variables will be qualitatively described as a collection of regions which are monotonically increasing with respect to some variables, decreasing with respect to others and constant with respect to the remainder. A tuple of values, say '+1' for increasing, '0' for constant and '-1' for decreasing, would be a suitable data structure for representing the qualitative description of each region of the function. Again an alternative approach would be to note that for a function of n variables there are three possibilities with respect to each variable and n independent variables which yields 3^n qualitatively different kinds of regions which could therefore be represented by a single integer between, say 1 and 3^n . Again the choice would seem to be determined by a compromise between storage requirements and ease (speed) of access.

However, these representation schemes do not take into account the problem of the flexibility of interpretation of geometric knowledge. 'Flexibility of interpretation' means that in practice the interpretation placed on any piece of geometric knowledge (see Section 3.5) may be changed dynamically at run time. Therefore, for a piece of geometric knowledge in an n -dimensional space the qualitative representation must describe how each of the n dimensions, considered as the range space of a function whose domain space is formed by the remaining $(n-1)$ dimensions varies qualitatively with respect to these domain dimensions. The union of all these qualitative descriptions provides the required flexibility of interpretation as the qualitative description of any relationship from $(n-r)$ dimensions into

the remaining r can simply be obtained by selecting the qualitative descriptions corresponding to these r dimensions.

Relationships which are not functions according to the mathematical definition (see Section 3.6), can be treated as a list of functions if the relationship yields a set of discrete values, or as a pair of functions (one each for the lower and upper bounds) if it results in an interval of values. The most general case is represented by means of a list of qualitative descriptions of either a single function or of pairs (that is a list of two elements) of functions.

The final constraint on the qualitative representation system follows from the ability to group detailed, small scale descriptions into larger units by means of the excursion box hierarchy. This will require qualitative descriptions of larger segments of relationships which are not monotonically increasing (decreasing or constant) functions. The qualitative reasoning vocabulary must therefore be itself increased to reflect these new cases. In order to determine the nature of the required additions, consider the simple case of a 'function' of a single variable which increases non-monotonically. Its graph (which can be represented in a two-dimensional space) will resemble a, possibly irregular, 'sawtooth' shape whose overall tendency is nevertheless increasing (that is the sum of the range intervals over which the graph increases is greater than the sum of the range intervals over which it decreases). The initial suggestion of a generalisation of the qualitative description of 'monotonically increasing function' could be chosen as 'non-monotonically increasing function' with parallel definitions for the 'decreasing' and 'constant' cases. This would, however, be an unwise choice. The reason is that the inverse of that function will be a relationship which is also non-monotonically increasing but is not necessarily a function. Therefore a better representation would be in terms of 'non-monotonically increasing (decreasing, constant) relationship' which would present one formalism both for cases that are proper functions and for those that are not.

From the point of view of performing deductions on the qualitatively described relationships, the representation ought to be taken one step further by adding an additional item of information to measure the 'amount' of non-monotonicity.

Consider the graph of a non-monotonically increasing function (or relationship) of a single variable. This graph will consist of an alternating sequence of increasing and decreasing sections in which the increasing sections dominate. Consider therefore placing an excursion box around each decreasing section and further consider the largest of these excursion boxes. This may be used as a guide to the amount of non-monotonicity; any movement in the independent variable whose magnitude is larger than that of (the projection of) this excursion box (onto the space of the dependent variable) will indeed result in an increase in the value of the dependent variable. Therefore, provided the proposed domain of operation is of sufficient magnitude, the relationship could in fact still be treated as monotonically increasing. The size of the largest 'non-monotonicity' enclosing excursion box would represent the limit of the validity of this assumption, below which the relationship would in fact have to be considered in more detail and a lower level of the excursion box hierarchy accessed. Similarly the size of this 'largest non-monotonicity' enclosing excursion box could also be used as a measure of the uncertainty of any deduction result obtained by assuming the represented relationship to be a monotonic function.

Indeed, it could be argued that such a mechanism is in fact already required for the representation of the geometric shape of engineering components. These are modelled as a continuously varying parameter, whereas in reality this assumption becomes invalidated at small scales when the crystalline, molecular or atomic structure of the underlying material is approached. The size of the largest 'non-monotonicity' is, however, so small compared to the size of the whole engineering component as to become sufficiently insignificant for the assumed 'continuity' to be used without restrictions by existing CAD systems.

However, all the above ideas remain at a very informal stage and require further research to verify that they are indeed correct. Should this be the case, the above scheme would represent a novel means of representing 'approximate' information with an integrated, objective limit as to its validity.

Further note that the above are the only extensions that appear to be required as the thus augmented qualitative descriptive vocabulary can now apparently

represent all situations that may arise in the representation of geometric knowledge.

However, it must be emphasised that both constructive definitions and qualitative representations are areas which still require more research and the above ideas will evolve as additional insights are gained.

The next Section will consider the implementation issues surrounding some of the geometric modelling algorithms needed for the proposed geometric knowledge representation system.

4.3 ALGORITHMS

This Section considers some of the geometric modelling algorithms for the generalised solid modeller which is needed for the representation of geometric knowledge.

The principal primitive operations are Join (or convex hull), Cartesian product and intersection. Most of the remaining modelling operations such as set theoretic union can in fact be expressed in terms of these primitives, or reduce to simple data structure access such as, for example, building a list of polytopes. The following Sections consider each of these algorithms in turn.

4.3.1 Convex Hull (Join) Algorithm

The Join (or convex hull) operation takes a set S of p points in a n -dimensional space and finds the smallest convex set which contains S (See Appendix A for the definition of terms such as convex).

A large number of convex hull algorithms exist [LEE.84]. Most concentrate on the computation of two-dimensional convex hulls and while there are a few which consider the three-dimensional problem, those for higher, n -dimensional cases are very much rarer.

Two- and three-dimensional convex hull problems can be computed in $O(p \log p)$ time where p is the number of points [PREP77]. In both cases these times are optimal, though, in the two-dimensional case they can be improved when additional information about the input points is known [LEE.83]. The computational complexity of convex hull problems in higher dimensions is more difficult to determine as it depends to some extent on the properties of n -dimensional polytopes. However, this area and in particular the upper bounds on the number of components of a polytope is complex and but incompletely understood [GRUN67 Chapter 10].

For example, the divide and conquer algorithm for two- and three-dimensional convex hulls described in [PREP77] is based on the property that the number

of edges of the convex hull with p extreme points (vertices) is at most linear in p . However, in higher dimensions there are polytopes whose number of (hyper) edges is $O(p^2)$ in the number p of vertices [GRUN67 Section 10.1] so that a generalisation of that particular method to higher dimensions is not possible within $O(p \log p)$ time.

In fact only one algorithm for computing convex hulls in n -dimensions was found [CHAN70]. No bounds on its computational complexity were given other than an empirical (favourable) comparison with the 'brute force' method. Nevertheless, an optimal algorithm for computing convex hulls in n -dimensional space where n must be an even number has been reported [LEE_84].

Therefore, for optimal performance, a mixture of algorithms should be used : for two- and three-dimensional problems use the corresponding optimal algorithms, for n -dimensional problems (where $n > 3$), if n is even use the algorithm referred to in [LEE_84] and if n is odd use the algorithm reported in [CHAN70]. This approach would ensure best known performance at the cost of programming complexity.

Alternatively, because of its generality, the algorithm described in [CHAN70] could be used throughout obtaining simplicity of implementation possibly at the expense of computational efficiency. However, this means that the discussion can be from now on be restricted to this single algorithm without compromising any functionality.

4.3.1.1 Finite Join Algorithm

In their paper [CHAN70] Chand and Kapur both supply a proof of correctness and state their convex hull (Join) algorithm as implemented in a FORTRAN program. This algorithm computes the convex hull of a finite set of finite points in n -dimensional space through a process of 'gift wrapping' in which the $(n - 1)$ -dimensional faces of the polytope are generated one at a time.

The algorithm relies on the property that exactly two $((n - 1)$ -dimensional) *faces* of an n -dimensional polytope P intersect along each $((n - 2)$ -dimensional) *edge* of P . If a face F and edge E belonging to F are known, the second face F' can be found by conceptually rotating the plane of the known face F about edge E through an appropriate angle. This process is similar to that of wrapping up a gift in a sheet of paper. The new face F' will be bounded by new edges (at least $n - 1$ of them). This process of gift wrapping can therefore be repeated until all edges are contained in exactly two faces at which stage the polytope has been fully determined. A mechanism for determining a starting face is also needed.

This 'gift wrapping' algorithm must be extended to include the case where some (or all) of the points are at infinity. For this purpose, points, both finite or at infinity, are represented by their homogeneous coordinates (see Section 4.2). The extension of the convex hull (Join) algorithm to points at infinity is not too complex but does introduce a number of additional cases which must all be taken into account.

The main convex hull or Join algorithm (as found in [CHAN70]) is set out below. Its statement differs somewhat from that found in [CHAN70] as it appeared more 'natural' to use the concept of recursion in the description of an essentially recursive algorithm.

The input to the algorithm is a finite set P of points in an n -dimensional space, the output is the convex hull of P .

The preparatory work includes two tasks.

The first is to attempt to simplify the Join task by operating in the lowest dimensional space possible. To this end a coordinate system for all points in P is calculated incrementally and the dimension d of the set P obtained ($d \leq n$). When d is less than the dimension n of the Join space, the Join task can indeed be simplified by working in the lower d -dimensional space. In this case the points of P are projected into its d -dimensional space, which is then used for the Join (convex hull) operations. However, the effect of this projection must be reversed before returning the final result.

The second preparatory task consists of filtering out cases where the dimension d of the set of points P is trivially low, such as zero-dimensional (where the convex hull of a single point is the point itself) or one-dimensional (where the convex hull reduces to finding the extrema of an interval).

Assuming henceforth that the dimension of the set of points P is greater than unity, the algorithm proceeds as a sequence of steps. The main concepts (which are reflected in the data structures) involved are that of an *edge* about which to 'gift wrap' the already known *parent face* of that *edge* and the set of all the other points in P .

FINITE JOIN ALGORITHM

STEP 1 : get an initial *edge*.

This routine assumes that there is at least one finite point. The goal is to obtain a subset of points of P which defines an initial (prototypical) *edge* which will form part of the boundary of some face of the final convex hull of P . An initial inward (into the convex hull) pointing unit vector $\vec{N}$ is also required. The initial subset of points of P consists of all points with minimum first coordinate value. The dimension of this subset is computed and the unit vector along the first coordinate axis is used as the required $\vec{N}$. Note that as there is (as yet) no *parent face* from which this prototypical *edge* could have been extracted, the set of points defining this *parent face* remains empty.

STEP 2 : hide any existing *edges*.

Hide any already existing $((n - 2)$ -dimensional) *edges* so as not to confuse these with the ones which will be created as part of the convex hull of P . Note that such *edges* can only exist as the result of earlier recursive computations.

STEP 3 : MAIN LOOP.

While all faces of convex hull have not yet been computed repeat steps 4 to 7.

STEP 4 : gift wrap a face.

Points from P which are neither in the *parent face* nor in the *edge* are added to the points defining the *edge* through (if necessary, repeated) 'gift wrapping' operations until it contains enough points to fully define a face F' of the convex hull of P . The inward pointing unit vector normal $\vec{N}$ to this face F' is also computed.

STEP 5 : find the boundary components of this new face F' .

If the new face F' contains more than (dimension of face + 1) points then F' is a non-simplectic polytope, otherwise it is a simplex. If F' is a simplex then the set of all its boundary components is simply the set of all (convex hulls) of subsets of points of F' . If F' is a non-simplectic polytope then apply the finite Join algorithm recursively to the $(d - 1)$ -dimensional set of points F' .

STEP 6 : keep a record of new *edges*.

The new face F' is bounded by edges. Some of these may already exist as the result of earlier computations whereas others are new. If an edge already exists, then both faces containing that edge have now been found and it need not be considered as a candidate 'gift wrapping' *edge*. If the edge is new then only one face containing it has been found and it is therefore eligible for a subsequent gift wrapping operation. In either case the status of each *edge* must be updated by marking it as *completed* if it already exists, *incomplete* if it has not been found earlier.

STEP 7 : select a next *edge* about which to gift wrap.

From amongst the computed edges one which is marked as *incomplete* must be found. However, only edges which contain at least one finite point are eligible as it is not possible to 'gift wrap' about an edge at infinity. If such an *edge* can

be found then it will be used as the 'hinge' about which STEP 4 can gift wrap, together with its *parent face* and inward pointing normal vector $\vec{N}$. If no such *edge* can be found then the (finite part of the) convex hull has been completely determined and the loop of STEP 3 terminates.

Note that the recursive call of the algorithm to itself terminates because the dimension of the problem decreases each time and one-dimensional convex sets (intervals) are simplectical.

The centre piece of this algorithm is clearly the 'gift wrapping' process which can itself be described as a sequence of steps.

Consider a n -dimensional space containing a $(n - 1)$ -dimensional face F which has an $(n - 2)$ -dimensional edge E as part of its boundary. The face F has a unit normal vector $\vec{N}$ which points into the set of points P whose convex hull is being computed.

GIFT WRAPPING ALGORITHM

STEP 1 : calculate the $\vec{E}$ vector.

This is a preparatory step in which a coordinate system (whose origin is a *finite* point of the edge E) is constructed starting with the normal inward vector $\vec{N}$ and using all the points of the edge E . When the points in E are exhausted, the coordinate system is still incomplete as only $(n - 1)$ dimensions have been specified. The required $\vec{E}$ vector is the vector which is orthogonal to all coordinate system vectors found so far and whose orientation is such that it points into the face F . So the required $\vec{E}$ vector can be obtained by "growing" another coordinate system vector towards a point of the face F which is not also in the edge E . During the startup period of the algorithm, that is before a first face has been calculated, the orientation of the $\vec{E}$ vector is not important as there are as yet no faces which might otherwise be found twice over. So, in this case the coordinate system is completed by "growing" as many further coordinate vectors as needed. In either case, the properties of the $\vec{E}$ vector are (1) $\vec{E}$ is orthogonal to the normal inward vector $\vec{N}$, (2) $\vec{E}$ is orthogonal to the dimensions of the gift wrapping edge E , and (3) the scalar product $\vec{E} \cdot \vec{V}_i$ is positive for at least

one point P_i in F (or P as F is empty during the startup of the algorithm). Note that $\vec{V}_i = \vec{OP}_i$ where O is the origin of the coordinate system.

STEP 2 : MAIN LOOP.

While the set of points defining the edge E , augmented by 'gift wrapped' points has not reached the dimension of a face repeat the 'gift wrapping' steps 3 to 5 so as to add still further points. In fact, the dimension of the set of points defining the edge E is increased by at least one by each 'gift wrapping' cycle. During the startup period of the algorithm (which ends when the first face has been obtained) several cycles may be required to obtain a first face. However, once the startup period is over, only a single gift wrapping cycle is needed to go from the set of points defining an $(n - 2)$ -dimensional edge E to a $(n - 1)$ -dimensional set defining a new face F' .

STEP 3 : find (gift wrap) points.

The supporting (hyper) plane through the edge E is rotated ('gift wrapped') about E , until it meets some points of P . These points are thereby identified as extreme points of the convex hull of P and are added to those defining the edge E to yield a set defining a new face F' .

The (two-dimensional) plane G defined by $\vec{E}$ and $\vec{N}$ is orthogonal to the edge E about which 'gift wrapping' occurs. The vectors $\vec{E}$ and $\vec{N}$ also form a coordinate system of G . Note that the 'gift wrapping' rotation occurs only within the two-dimensional plane G as all other dimensions are in the rotation invariant edge E . Therefore even though the convex hull problem operates in an n -dimensional space, the computations need only take place within an appropriately chosen two-dimensional plane.

Consider all points P_i of P which are neither in the edge E nor its parent face F . When these points P_i represented by a coordinate tuple $\vec{V}_i$ with respect to the 'gift wrapping' coordinate system are projected onto the plane G the coordinates of their image within G are $(\vec{V}_i \cdot \vec{E}, \vec{V}_i \cdot \vec{N})$. The angles θ_i that the (projected) $\vec{V}_i$ make with the $\vec{E}$ axis range from 0 for a (projected) point on the $\vec{E}$ axis through 90° for a (projected) point on the $\vec{N}$ axis up to 180° for a (projected)

point along the $-\vec{E}$ axis. That values of the angles θ_i outside the range $]0^\circ, 180^\circ[$ cannot occur follows from the definition of the points P_i and the fact that the $\vec{N}$ points towards the interior of the convex hull of P .

The points sought after by the 'gift wrapping' process are those which have the greatest angle θ_{max} with the $\vec{E}$ axis. In the conceptual rotation of the (hyper) plane containing the known face F about the edge E the points with corresponding angle θ_{max} will be encountered first. These points are therefore the required extreme points which (when added to the points defining the edge E) will define the new face F' . This is the core of the process of "gift wrapping".

So for each point P_i the corresponding value of the angle θ_i could be computed and the relevant points identified. However, this would involve rather computationally expensive inverse-trigonometric operations. Instead the decision can be made in terms of a function of θ_i which can be computed easily and which is monotonically increasing with θ_i over the range $]0^\circ, 180^\circ[$. Such a function is provided by minus the cotangent of θ_i which can be computed by :

$$-\cotan(\theta_i) = -\frac{\vec{E} \cdot \vec{V}_i}{\vec{N} \cdot \vec{V}_i}$$

The 'gift wrapping' process can finally be described. For each point P_i of P which is not in the edge E or the face F do the following two steps, one computational and the other making a decision.

The computation is to calculate the value of $-\cotan(\theta_i)$. This requires first obtaining the vector starting at the origin O of the gift wrapping coordinate system and ending at P_i . If the point P_i is a point at infinity represented by its homogeneous coordinates, then simply interpret the first n values as a vector pointing towards this P_i and use this as $\vec{V}_i$. If P_i is a finite point, then use $\vec{V}_i = \vec{OP}_i$. Finally use the above equation to obtain the value of $-\cotan(\theta_i)$.

The decision involves comparing the value of $-\cotan(\theta_i)$ with the current maximum $-\cotan(\theta_{max})$ and taking an appropriate action.

$$\text{if } \left\{ \begin{array}{ll} -\cotan(\theta_i) < -\cotan(\theta_{max}) & \text{Proceed with next point.} \\ -\cotan(\theta_i) = -\cotan(\theta_{max}) & \text{Add point } P_i \text{ to the set of kept points.} \\ -\cotan(\theta_i) > -\cotan(\theta_{max}) & \text{Discard current set of kept points, use current point } P_i \text{ as the first element of a new current set of kept points and update current value of } -\cotan(\theta_{max}). \end{array} \right.$$

STEP 4 : update the set of edge points E .

Add the points found by the 'gift wrapping' process to those defining the edge E and calculate the augmented dimension of the resulting set. Also update the inward pointing vector using the following equation and rescaling to unit length.

$$\overrightarrow{N_{new}} = -\cotan(\theta_{max})\overrightarrow{N_{old}} + \overrightarrow{E_{old}}$$

STEP 5 : recalculate $\overrightarrow{E}$.

If the dimension of the augmented set of points has not yet reached $(n - 1)$ then a new face has not yet been found and another 'gift wrapping' cycle is needed. In this case the $\overrightarrow{E}$ must be recalculated using the method of STEP 1.

This completes the description of the finite Join (convex hull) algorithm. The next section now considers the extension of this algorithm to the case where some, or all, of the points are at infinity.

4.3.1.2 Infinite Join Main Algorithm

The algorithm described in the previous Section is in fact already slightly modified from the original of [CHAN70] as the idea of points at infinity is mentioned twice (once during the selection of the next edge about which to 'gift wrap' and once in the 'gift wrapping' itself). In fact this does not so much represent an extension of the algorithm as an indication of how it can be made to cope with the case of finding the (finite part) of the convex hull when there are both finite points and points at infinity to deal with.

If P is again a finite set of points whose convex hull is sought, the complete range of cases to be considered are : " P contains only finite points and no points at infinity", " P contains no finite points and only points at infinity", " P contains both finite points and points at infinity" and finally " P is empty".

When P is empty so is its convex hull. When P only contains finite points the algorithm of the previous section can be used directly. This leaves two new cases where P contains points at infinity with or without containing finite points as well.

Because of the difficulty discussed in Section 4.2 resulting from the use of homogeneous coordinates in relation to the representation of whole linear spaces the possibility that the convex hull might include a linear space must be investigated.

Therefore the highest level of the augmented convex hull algorithm is organised as a sequence of two steps, the first computational and the second a decision making one. The computational step consists in determining the dimension d_L of the largest linear space L contained in the Join of the points at infinity. The decision step consists of taking an action according to the value of d_L as set out below (Note that n is again the dimension of space containing P) :

if	{	$d_L = n$	The Join (convex hull) of P is equal to the whole space, so create a polytope (for example in the shape of an infinite excursion box) corresponding to the whole space.
		$d_L = 0$	The Join (convex hull) of P will not contain any proper linear spaces, so use an augmented 'infinite' Join algorithm to compute it.
		$0 < d_L < n$	The Join of points contains a proper linear space L , so proceed as follows. Project all points of P onto the linear space $L^\perp$, orthogonal to L . (Note that this projection may result in some infinite points of P becoming finite). Depending on the presence of points at infinity, use either the finite Join or the augmented 'infinite' Join algorithm with this projected set of points P' . Finally take the Cartesian product of the convex hull of P' with the linear space L to obtain the convex hull of P .

Although this overcomes the problems arising out of the use of homogeneous coordinates, it does introduce the need for additional algorithms. Calculating the linear space included in the convex hull of points at infinity and computing the 'infinite' convex hull (where some of the points are at infinity but do not include a proper linear space) are investigated next.

4.3.1.3 Contained Linear Space Algorithm

The Join (convex hull) of a set of points at infinity may be equal to, or contain a complete linear space L . If it does not the Join reduces to a single convex polytope at infinity. To determine which of the three cases, “contains a linear space L ”, “is equal to a linear space L ” and “does not contain a (proper) linear space” has occurred, only the points at infinity need to be considered.

As any convex hull (in particular, of the points at infinity) is homeomorphic to the unit ball of same dimension [STOE70 p124], the points at infinity can in fact be mapped onto the unit ball without loss of information. This effect corresponds to the view of an ‘infinite observer’ to whom all finite separations between points appear so small as not to be distinguishable whereas infinite separations appear finite. Such an ‘infinite view’ of the points at infinity may be obtained by a process resembling a hypothetical division by infinity which maps all finite points onto the origin (or indistinguishably close to it) and the infinite points onto the unit sphere. Thus the change corresponds to taking a ‘view from infinity’.

A point P_i at infinity is represented by homogeneous coordinates which may be interpreted as a vector $\vec{V}_i$ starting at the origin, ‘pointing’ towards P_i and thus defining it. Mapping a point at infinity P_i into a point P'_i at a unit distance from the origin simply involves rescaling the vector $\vec{V}_i$ to unit length and using the resulting coordinate tuple as the coordinates of the finite point P'_i .

If the convex hull of these finite points P'_i contains the origin then the convex hull of the original points at infinity P_i will contain a linear space. The dimension d_L of the *component* of the finite convex hull (either the interior, or some boundary face, edge, etc) which contains the origin is that of the linear space included in the Join (convex hull) of the infinite points P_i .

By computing a coordinate system whose first d_L dimensions span the origin containing component the contained linear space L is itself defined.

The algorithm for computing the linear space contained in a Join (convex hull) of points at infinity consists of the following sequence of steps.

CONTAINED LINEAR SPACE ALGORITHM

STEP 1 Convert the points at infinity into finite points.

STEP 2 Calculate the (finite) convex hull of these points.

STEP 3 Find the *component* (if any) of this Join (convex hull) which contains the origin.

STEP 4 Calculate a coordinate system for this d_L -dimensional component.

STEP 5 Restore the points at infinity.

The most complex step in the above list is clearly finding the *component* C (if any) of a Join (convex hull) J which contains a given (finite) point O . This step is now described in more detail below.

Consider the (hyper) plane H which passes through the point O and is perpendicular to the last coordinate axis. Some 'slice' S of the Join (convex hull) J will be included in H . As the intersection of two polytopes J and H , S is also a polytope.

Each component of the 'slice' S is related to some component of the Join J by virtue of the fact that each component of S is the 'cut' of some component in J . These correspondences between components of J and S are stored.

If S is the empty set then there is no intersection, and therefore no component containing O .

If S is a point then there is no need for further action as either the point S is equal to O or it is not. In the first case the required component is the antecedent of S in J from the (possibly repeated) intersections with (hyper) planes. In the second case there is again no intersection, and hence no component containing O .

If S is of dimension greater than zero, S is a proper polytope, and the 'slicing' algorithm must be applied recursively to S within its containing space H . Note

that the implied projection of S into H is achieved simply by ignoring the last coordinate values of the points of S (with only a minor adjustment being required for the homogeneous coordinate). This process terminates as the dimension of the polytope S decreases with each level of recursion.

Computing the intersection of a polytope J with a (hyper) plane H can itself be achieved by considering all the *components* of J in order of increasing dimension.

The zero-dimensional components (vertices) of J can be classified by using the total ordering along the coordinate axis perpendicular to H as either below H , included in H or above H .

If all the vertices are not strictly on the same side of H then J and H do intersect yielding a “slice” S which is computed as follows.

Consider each one-dimensional edge of J and, using the positioning of its two end points, determine whether it is cut by H (the end points are on opposite sides of H), disjoint from H (the end points are on the same side of H) or included in H (the end points are both included in H).

Now consider the higher, i -dimensional components of J and similarly use their $(i - 1)$ -dimensional faces and $(i - 2)$ -dimensional edges to determine whether they are either included in (all faces are included) or disjoint from (no edge or face is cut and at most one edge or face is included) or cut by (all other cases) H .

The above process will fully determine all components of the intersection S of J and H . The next Section now considers the Join (convex hull) algorithm when points at infinity are present.

4.3.1.4 Infinite Join Algorithm

This section describes the ‘infinite’ Join algorithm which computes the convex hull J of a set of points P (in some n -dimensional space) which comprises at least one point at infinity, zero or more finite points and does not contain any (one-dimensional or greater) linear space. This means that the convex polytope J will only have as single component (either a $(n - 1)$ -dimensional face, or an $(n - 2)$ -dimensional edge, or ..., or a point) at infinity. The infinite Join algorithm therefore consists of the following sequence of steps.

INFINITE JOIN ALGORITHM

- STEP 1 Calculate the entity at infinity as well as a vector perpendicular to it.
- STEP 2 If there are any finite points then proceed with steps 3 to 6 otherwise the convex hull computation has already been completed.
- STEP 3 Change the coordinate system to one where the first coordinate axis is aligned with the normal to the entity at infinity. This ensures that the finite Join algorithm will find an initial edge containing at least one finite point to start the ‘gift wrapping’ operations.
- STEP 4 Use the finite Join algorithm of Section 4.3.1.1 to find the finite and trans-finite (which contain both finite and infinite points) components of J .
- STEP 5 Reverse the change of coordinate system of step 3.
- STEP 6 Add component at infinity to the finite and trans-finite components to yield the complete convex hull J of the set of points P .

The only part of the infinite Join algorithm which needs further description is the algorithm by which the component at infinity is obtained.

Its objective is to compute the component at infinity of a Join (convex hull) of points some, but not necessarily all, of which are at infinity. The component at infinity is at most $(n - 1)$ -dimensional as it lies in the ‘boundary’ of the n -dimensional (infinite) polytope corresponding to n -dimensional space. The

algorithm for computing this component at infinity can again be described as a sequence of steps.

COMPONENT AT INFINITY ALGORITHM

- STEP 1 Convert each point at infinity P_i into a finite point P'_i using the method described in Section 4.3.1.3.
- STEP 2 Add the origin point O of the coordinate system of the n -dimensional space to the set of points P'_i .
- STEP 3 Calculate the Join (convex hull) J of this set of finite points as well as its dimension d_J .
- STEP 4 Discard all components of J which contain the origin point O . The list of discarded components includes the d_J -dimensional interior of J as well as all $(d_J - 1)$ -dimensional faces containing $(d_J - 2)$ -dimensional edges, ..., containing one-dimensional edges which contain the origin point O as well as the point O itself. The remainder forms a connected set of k -dimensional components which are exactly those boundary components which are at infinity.
- STEP 5 If there are several of these k -dimensional components at infinity, then merge them through elimination of duplicate (interior) edges and fusion of adjacent faces. This results in a single k -dimensional, finite equivalent of the component at infinity.
- STEP 6 Restore the infinite points to infinity and in the process obtain the component at infinity.
- STEP 7 Calculate a (the) normal vector to the component at infinity. This is obtained by taking the vector sum of all (unit) vectors pointing towards the vertices of the component at infinity.

This finally completes the description of the augmented 'infinite' Join algorithm which can compute the convex hull of a set of points (finite or at infinity) in an n -dimensional space. The next section considers the Cartesian product operation which is required both as part of the Join algorithm and as a modelling operation in its own right.

4.3.2 CARTESIAN PRODUCT ALGORITHM

The Cartesian product operation takes a list of geometric models and constructs a new geometric model, corresponding to their Cartesian product, within a specified higher dimensional space (whose dimension must equal the sum of the dimensions of the operand models' spaces).

The Cartesian product command takes as operands a list of geometric entities as well as a set of *control words*.

The list of geometric entities contains the Cartesian product space $\Sigma_{\times}$ followed by m Cartesian product operands G_i (where $1 \leq i \leq m$) each of which is either a polytope, or a point, or a list of these.

The *control words* specify how the dimensions of each operand space map onto the dimensions of the product space $\Sigma_{\times}$. Each control word (one for each operand) consists of a string of digits with exactly one digit for each dimension of the corresponding operand's space. The digits range, for example, from 1 up to the dimension $d_{\Sigma_{\times}}$ of the product space inclusively. Taken over all control words, all the digits in that range must occur once and only once. The meaning of the digits is as follows. If there is a digit j in the i th position of the k th control word, then the i th coordinate of the of the k th operand model's points will be used as j th coordinate of the points of the resulting product model. This system allows full flexibility for the possible permutation of coordinates in the resulting product model.

The first stage of the Cartesian product algorithm is the highly recursive, systematic 'unpacking' of any geometric models consisting of lists of polytopes so as to obtain more primitive tuples of operands which only comprise individual polytopes (as opposed to list of polytopes). The Cartesian product of each of these more primitive operand tuples can then be evaluated separately and the results combined correspondingly.

Computing the Cartesian product $J_{\times}$ of a tuple of m polytopes J_i (where $1 \leq i \leq m$) is achieved by the following sequence of steps.

CARTESIAN PRODUCT ALGORITHM

STEP 1 : MAIN LOOP.

Repeat steps 2 to 8 for all dimensions d_x (from 0 up to the product space dimension d_{Σ_x}) of components of the Cartesian product polytope.

STEP 2 : find first operand dimension tuple.

Find the *first* tuple of m dimensions $d_{C_{J_i}}$ (where $1 \leq i \leq m$) of components C_{J_i} of the operand polytopes J_i which add up to the current value of d_x . Each value of $d_{C_{J_i}}$ occurring in the i th position of the tuple is restricted to the range 0 up to dimension of i th operand polytope J_i . The ordering for deciding which is the *first* tuple is based, for example, on the rule that the left-most elements of the tuple are those which change the least quickly or often as the set of all possible tuples is enumerated.

STEP 3 : operand dimension tuple loop.

While there is another valid operand dimension tuple, repeat steps 4 to 8.

STEP 4 : find first operand component tuple.

Find the *first* tuple of m components C_{J_i} (where $1 \leq i \leq m$) such that each component C_{J_i} is of dimension $d_{C_{J_i}}$ and comes from the i th operand polytope J_i . The ordering for deciding which is the *first* tuple is based, for example, on the rule that the left-most elements of the tuple are those which change the least quickly or often as the set of all possible tuples is enumerated.

STEP 5 : operand component tuple loop.

While there is another valid operand component tuple, repeat steps 6 and 7.

STEP 6 : Cartesian product.

Compute the d_x -dimensional component C_x of the Cartesian product polytope J_x which results from the Cartesian product of the polytopes C_{J_i} in the operand component tuple. If d_x is equal to zero then C_x is a point, otherwise it is a proper polytope.

If C_x is a point then its coordinates can be obtained from a suitably ordered union of the coordinates of the operand points C_{J_i} . If one of the operand points C_{J_i} is a point at infinity then C_x is also a point at infinity, otherwise C_x is finite.

When C_x is a polytope of dimension d_x its bounding $(d_x - 1)$ -dimensional faces must be found. Because the Cartesian product algorithm builds up the overall result components in the order of increasing dimensions the required faces have in fact already been calculated.

Thus the appropriate ones need *only* be accessed. However, the only access “key” available is that these faces were computed by the Cartesian product of all but one of the C_{J_i} where the ‘one’ was replaced by one of its own faces. Thus the boundary of C_x can be found by considering each position i (where $1 \leq i \leq m$) in turn, further considering each face $F_{C_{J_i}}$ of the operand C_{J_i} at the i th position, (provided that there are any, that is provided that C_{J_i} is not a point) and finding the Cartesian product component corresponding to the operand component tuple in which C_{J_i} is replaced by $F_{C_{J_i}}$. The list of all faces thus generated forms the boundary of the product C_x .

STEP 7 : Next operand component tuple.

Find the *next* tuple of m components C_{J_i} (where $1 \leq i \leq m$) such that each component C_{J_i} is of dimension $d_{C_{J_i}}$ and comes from the i th operand polytope J_i . When no next tuple can be found, leave the loop of step 5.

STEP 8 : Next operand dimension tuple.

Find the *next* tuple of m dimensions $d_{C_{J_i}}$ (where $1 \leq i \leq m$) of components C_{J_i} of the operand polytopes J_i which add up to the current value of d_x . When no next tuple can be found, leave the loop of step 3.

STEP 9 : correct the Cartesian product.

The Cartesian product was computed as though all points were in finite space. However, when this is not the case and there are points at infinity the above algorithmic steps lead to duplicate Cartesian product points : Cartesian products of different finite points (which are therefore finitely separated) with a same point

at infinity will all yield the same product point at infinity. This is caused by the use of homogeneous coordinates as discussed in Section 4.2.1. These duplicate points will in turn lead to higher dimensional components with duplicate boundary elements, or even spurious higher dimensional components. In fact such spurious higher dimensional components consist only of a single point at infinity, which, when seen by a finite observer, appears as a whole higher dimensional component.

The Cartesian product therefore must be corrected by eliminating all the duplicate product points and their associated higher dimensional components.

This is achieved in three stages. First all duplicate points are found and marked as such. Note that a point is duplicated when there is another point with the same coordinates. Secondly the duplicate point information is propagated to all higher dimensional components whose spurious boundaries are identified and marked. Finally when all redundant points and components have been clearly marked and they can be discarded.

This completes the description of the Cartesian product algorithm. As illustrated by the above detailed description, the main complexity of the Cartesian product operation resides in the number of concepts and entities involved and in particular the sometimes complex manner in which these have to be accessed. Indeed, in order to minimise the complexity of access to previously computed components, the generation of duplicate and spurious information was permitted despite the need for a subsequent step to eliminate them.

The next Section describes the set theoretic operations of intersection and set-difference.

4.3.3 SET THEORETIC ALGORITHMS

The set theoretic operations used in geometric modelling are intersection, set-difference and union. These are defined in the usual manner as follows. Considering two geometric models M_1 and M_2 , their intersection $M_{M_1 \cap M_2}$ is the set of points common to both M_1 and M_2 , their union $M_{M_1 \cup M_2}$ is the set of points which occur in at least one of M_1 , M_2 and the set theoretic difference $M_{M_1 - M_2}$ is the set of points which belong to M_1 but do not belong to M_2 .

The first stage of the set theoretic operation algorithms is the highly recursive, systematic 'unpacking' of any geometric models consisting of lists of polytopes so as to obtain more primitive tuples of operands which only comprise individual polytopes as opposed to a list of polytopes. The selected set theoretic operation can then be applied individually to each of these more primitive operand tuples and the results combined afterwards. In the case of an intersection or set theoretic difference operation an inexpensive excursion box based comparison can be used for early identification of operands which do not intersect (by virtue of their excursion boxes being disjoint) and thus avoid the computationally more expensive algorithms outlined below.

The intersection $J_{J_1 \cap J_2}$ and set difference $J_{J_1 - J_2}$ of two polytopes J_1 and J_2 of same dimension d can be computed through the following sequence of steps.

INTERSECTION AND DIFFERENCE ALGORITHM

- STEP 1 initialise two variables J_- and $J_\cap$. J_- is the cumulative result for the set difference $J_1 - J_2$ and is initially equal to the empty set and $J_\cap$ is the current upper bound on the intersection of J_1 and J_2 and is initially equal to J_1 .
- STEP 2 repeat steps 3 to 6 for each face F_{J_2} of J_2 which contains at least one finite point. Faces at infinity can be ignored as they do not contribute any information, mainly because faces at infinity lie in the boundary of the whole space : As everything inside the space is necessarily on the 'inside' of its boundaries, no information can be eliminated as a result of applying steps 3 to 6 to a face at infinity.

- STEP 3 construct the support hyper plane H containing F_{J_2} .
- STEP 4 use the algorithm of Section 4.3.1.3 to cut $J_\cap$ with H .
- STEP 5 keep the portion (if any) of $J_\cap$ which is on the same side of H as J_2 and use this as the new value of $J_\cap$.
- STEP 6 if the set difference $J_1 - J_2$ is to be computed add the portion (if any) of $J_\cap$ which is on the opposite side of H to J_2 to the cumulative total J_- .

The above algorithm can also be applied to two operands of different dimension. In this case the task may first be simplified by operating in the space of the lower dimensional operand after intersecting the higher dimensional operand with this space. The intersection of an operand with a linear space L can be achieved by successive intersections with an appropriate (hyper) plane (see Section 4.3.1.3) to eliminate the additional dimensions one at a time. This process results in two polytopes of the same (lower) dimension to which the above algorithm can be applied.

The extension to the intersection of multiple polytopes is also simple : the intersection of the first two polytopes is computed using the above method and the result then intersected with the next polytope, and so forth until the intersection of all polytopes has been obtained.

Though not necessarily very efficient the algorithm outlined above has the advantage of both simplicity and independence of dimension and effectively deals with both intersection and set theoretic difference.

The set theoretic union of polytopes can also be obtained from the above algorithm by observing that the union of two polytopes J_1 and J_2 is equal to the list $J_1 - J_2$, $J_2 - J_1$ and $J_1 \cap J_2$.

The next Section describes the orthogonal projection operation.

4.3.4 ORTHOGONAL PROJECTION

The orthogonal projection operation takes a list of geometric models and constructs a new geometric model, corresponding to their orthogonally projected image, in a given lower-dimensional space.

The orthogonal projection command takes as operands a list of geometric entities as well as a single control word.

The list of geometric entities contains the projection space followed by the orthogonal projection operands each of which is either a point, a polytope or a list of these.

The control word specifies which of the dimensions of the space containing the operands are equivalent to the projection space. The control word consists of a string of characters, exactly one for each dimension of the operand space. The characters are either the underline character ('_') or a digit in the range from '1' up to the dimension of the projection space inclusive. All the digits in that range must occur once and only once. All other characters must be underline characters. The meaning of characters in the control word is as follows. If the character in the i th is a digit k then the i th dimension of the operand space will project into the k th dimension of the projection space. If the character in the i th is an underline character then the i th dimension of the operand space is orthogonal to the projection space and will therefore not be used. Note that the control word allows full flexibility for the possible permutation of coordinates in the resulting projected model.

Computing the orthogonal projection of a geometric model depends on its nature, that is whether a point, a polytope or a list of geometric models is being projected.

Starting with the case of an individual point, the projection algorithm consists of first creating a new projected point in the projection space and then determining its coordinates. In particular, for each dimension of the projection space, the appropriate value from the original coordinate tuple in the operand space is

selected using the information contained in the control word. This process will always map finite points into finite points, but may transform some points at infinity into the (finite) origin point of the coordinate system of the projection space. This phenomenon only occurs when a point at infinity lies in the linear space which contains the origin of the operand space and is orthogonal to the dimensions of the operand space which map into the projection space. This phenomenon is again the result of using homogeneous coordinates to represent points at infinity (see Section 4.2.1).

When the operand is a polytope, the result is also a polytope. In particular the property that the Join (convex hull) of the projected extreme points of a polytope is equal to the projection of the initial polytope itself can be (and in the present case was) used to obtain a simple implementation of the orthogonal projection algorithm. Therefore, in the case of the projection of a polytope, only its extreme points are first projected and then their convex hull computed using the Join algorithm described above. Note that although this process is simple to implement (given the availability of the Join operation) it is not necessarily as efficient as a general algorithm for projecting a j -dimensional polytope contained in a d -dimensional space into a p -dimensional space ($j \leq d$ and $p < d$).

When the operand is a list of geometric models its orthogonal projection is equal to the list of the projections of its elements. This is built up recursively when an element of the list is itself a list of further geometric models.

This effectively completes the description of the important primitive modelling operators. However some further algorithms are very briefly considered in the following Section.

4.3.5 OTHER ALGORITHMS

This Section briefly considers the issues of excursion boxes, and constructive definitions.

The result of Appendix B suggests that the comparison of two n -dimensional excursion boxes can in fact be decomposed into n comparisons of the excursion intervals along each coordinate axis.

If the only motive for comparing two n -dimensional excursion boxes is to determine whether they are disjoint or not then the following method may be used. The excursion boxes are disjoint if at least one pair of excursion intervals (along any of the coordinate axes) is disjoint, and a pair of intervals $[Min_1, Max_1]$ and $[Min_2, Max_2]$ is itself disjoint if $Min_1 > Max_2$ (where Max_2 is finite) or $Min_2 > Max_1$ (where Max_1 is finite).

The second point about excursion boxes concerns the effects on the primitive operations in terms of excursion boxes. In particular, the excursion box of the Join (convex hull), union (or list) of two or more excursion boxes, is simply the least excursion box containing all the excursion boxes of the operands. The excursion box of a Cartesian product is the Cartesian product of the excursion boxes of the operands. Similarly, the excursion box of an orthogonal projection is the orthogonal projection of the operand's excursion box. The excursion box of an intersection is contained in, but may be smaller than (or even be empty), the intersection of the excursion boxes of the operands. Similarly, the excursion box of a set difference will be contained in the excursion box of the first operand.

Finally, constructive definitions can be implemented without great difficulty provided that the main program of the proposed system contains an expression parser and interpreter for managing the user interface. (This should not be interpreted to mean that the interface presented to the user would actually take the form of entering lines of command words or expressions from a keyboard). A relatively simple modification permits the same expression parser and command interpreter to accept its input from a data structure (for example, an array of characters) which can be stored as part of a constructively defined geometric entity

and which specifies its method of construction. This approach would have the advantage of providing equal power of expression to the user and constructively specified models while minimising the amount of programming effort.

The final Section of this Chapter will now provide a brief summary of the implementation issues raised so far.

4.4 IMPLEMENTATION SUMMARY

The algorithms described so far were mainly devised to enable the early development of a prototype system for the representation of geometric knowledge. For this reason the main emphasis was placed on obtaining a description suitable for implementation rather than conducting detailed computational complexity analyses (which may be difficult in cases where these depend on the complexity of the component structure of higher dimensional polytopes) or deriving rigorous proofs of correctness.

Most of the algorithms described so far in this Chapter were actually implemented. However, owing to the complexity of some of the algorithms it was not possible to implement a system for the representation of geometric knowledge or even complete the generalised n -dimensional polyhedral solid modelling system in the time available. However, the implementation as it currently stands will now be reviewed by considering in turn the choice of programming language, the organisation of the main program, the implemented algorithms and the choice of data structures.

The 'C' programming language was used for the implementation as it offered flexibility and a good programming environment. The modern functional programming language 'HOPE' [BAIL85] would have formed a preferred implementation language mainly because of its increased expressive power but could not be used as the language is still evolving and as yet only available as a demonstration prototype rather than a fully working system.

The implemented program effectively falls into three major sections. The first is the main program which consists of a command interpreter, the second the geometric modelling routines whose algorithms were described in Section 4.3 and the third the routines associated with manipulating the geometric model data structures.

The command interpreter which forms the main program takes a string of characters forming sets of key words and compares them to a list of known commands. When a match is found the associated routine is executed. When a command

requires additional parameters these are either supplied explicitly and follow the key word of the command, or apply implicitly to the geometric model which forms the current focus of attention. Currently commands are obtained either from the keyboard or a file, and are stored as in an array of characters. This organisation was used in anticipation of constructively defined geometric entities whose constructive definition could itself be stored in an array of characters and thus passed directly to the command interpreter when needed. However, constructively defined geometric entities would also have required a more sophisticated, possibly stack based method of passing geometric models as parameters into modelling commands. The current implementation does not support constructively specified geometric models. Furthermore, the current user interface is limited to giving commands via the keyboard as there was no time to implement any form of graphical output or interface.

With the exception of the intersection/set difference algorithm which could not be completed in time, all algorithms of Section 4.3 were implemented as described. Not surprisingly the Join (convex hull) algorithm was by far the most complex algorithm to implement with approximately 6000 lines of code with the Cartesian product operation following at around 2000 lines. Numerous tests were run on all algorithms in order to ascertain their correctness under various input conditions which included changing the kinds of points present (the respective absence or presence of finite points and points at infinity) and the dimension of the space (up to 10 dimensions for computing the Join (convex hull) of the vertices of a simplex). In all test cases the algorithms performed as expected, providing the correct answers.

The suggested data structures for the representation of geometric knowledge as outlined in Section 4.2 are mainly the result of experience gathered during the implementation of the main modelling algorithms while using a less fortunate choice of data structures. In particular, in the initial stages of the implementation the desire to avoid the complexity of dynamic data structures led to a fixed, array based representation which was subsequently had to be supplemented by dynamic data structures. The resulting dual data structures then required additional routines to translate between them. The overall effect is therefore neither very

efficient nor elegant. The main cause of these problems was a lack of appreciation of the increased structural complexity of polytopes in higher dimensions and the need for more flexible access which resulted from it.

Even though the implementation could not be completed, it has still served a very useful purpose in exposing the problem areas and clarifying the requirements that the data structures for storing geometric knowledge have to support in terms of flexibility of access. The scale of the programming effort needed to implement even a basic system has also become apparent as the (commented) source code currently stands at around 17000 lines.

This concludes the investigation into implementation issues. The next Chapter will now illustrate the concept of geometric knowledge.

Chapter 5. GEOMETRIC KNOWLEDGE EXAMPLES

This Chapter illustrates the concept of *geometric knowledge* by means of some simple examples. However, this process is necessarily subject to certain difficulties in the absence of a fully implemented system as these examples cannot actually be demonstrated. The examples considered in the following Sections are the representation and use of two particular geometric models (the addition plane and the multiplication surface), an illustration of the excursion box hierarchy for intersection computations and a final brief outline of (future) 'geometric reasoning'.

The purpose of these examples is not only to illustrate the modelling operations but also the multiple interpretations that may be placed on a geometric model which thereby becomes a piece of *geometric knowledge*.

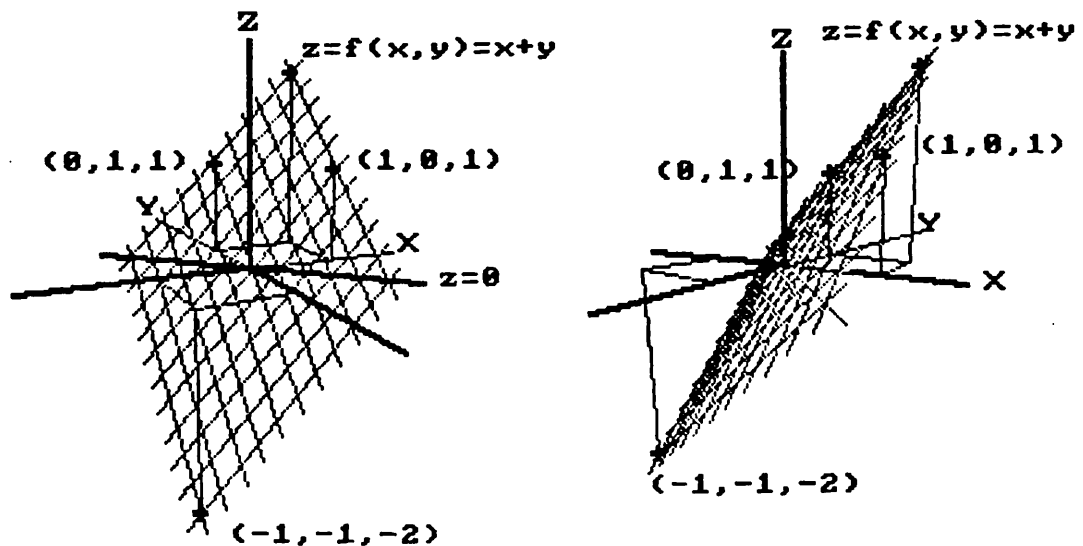


Figure 5.1. The addition plane.

5.1 THE ADDITION PLANE

This example of geometric knowledge consists of creating and using a geometric model which happens to be the graph of the addition function $z = f(x, y) = x + y$ as shown in Figure 5.1.

The graph of the addition function $z = Add(x, y) = x + y$ is simply a 'tilted' plane in a three-dimensional 'addition' space whose coordinate axes correspond to the x , y and z values. This 'addition' plane could be represented in the proposed modelling system by a single polytope obtained as the join of three points at infinity, for example, $(1, 0, 1, 0)$, $(0, 1, 1, 0)$ and $(-1, -1, -2, 0)$ in homogeneous coordinates. As the addition plane actually contains the origin, such a representation would indeed be possible in spite of the use of homogeneous coordinates (see Section 4.2.1). Alternatively, it could be represented as a list of, for example, four polytopes each of which formed a quadrant of the addition plane. In fact, in the implemented system this is the representation that is actually used as, by default, all n -dimensional linear spaces (and a plane is a two-dimensional linear space) are represented as a list of 2^n sectors.

Instead of being restricted to the interpretation of this model of the addition plane as a purely geometric model, it can also be considered as the representation of functional relationships. This will now be illustrated.

The additional concepts needed for the interpretation as a function are a domain space, a range space and some specific parameter in the domain space to which the function is to be applied. The restriction on the choice of domain and range spaces are that they must partition into two parts the dimensions of the space in which the graph of the corresponding relationship is represented. In the present case the addition space is three-dimensional, and thus there are only two non degenerate cases : the domain space is two-dimensional and the range space one-dimensional, or vice versa.

First consider the case where the domain is two-dimensional and the range therefore one-dimensional. There are again alternative mappings from the domain

space into the addition space depending on whether the domain space corresponds to the xy , xz or yz coordinate plane of the addition space. The range space would then respectively correspond to the z , y and x coordinate axis of the addition space. Whichever of these possibilities is chosen the image of a point in the domain is constructed using the method described in Section 3.5 :

Consider a point P_D in the two-dimensional domain space D with, for example, coordinates $(3.5, -2.0)$. Assume further that, for example, the domain space D corresponds to the xy coordinate plane of the addition space. The image point P_R of P_D in the one-dimensional range space R is constructed by the following sequence of steps (see Figure 5.2) :

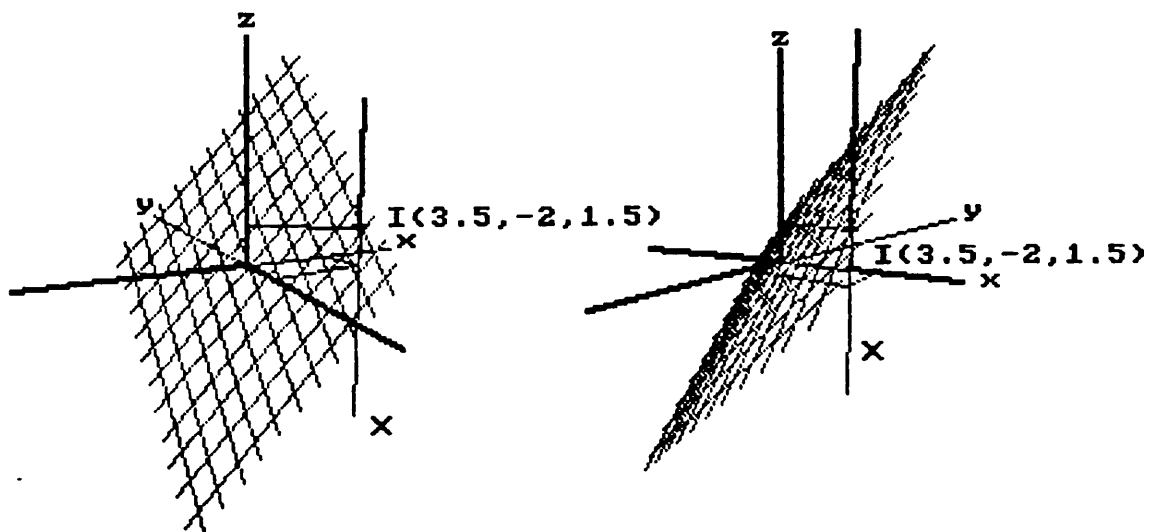


Figure 5.2. Applying the addition function.

MAPPING APPLICATION ALGORITHM

- STEP 1 In the addition space, form the geometric model X obtained by taking the Cartesian product of P_D with the range space R . X is the line parallel to the z axis which contains all points whose coordinates are of the form $(3.5, -2.0, \lambda)$ where λ is any real number.
- STEP 2 The addition space now contains two geometric models, one representing the addition plane and the other the line X . Compute their intersection I . I is a single point with coordinates $(3.5, -2.0, 1.5)$.

STEP 3 Project this point I from the addition space onto the one-dimensional range space R where it yields the point P_R with coordinate (1.5). This is the image of P_D under the addition function, that is the sum of 3.5 and -2.0 .

If the two-dimensional domain space D had been chosen to correspond to the xz coordinate plane of the addition space, and the above sequence of steps applied, the result would change as follows. In step 1 the geometric model X would form a line parallel to the y coordinate axis of the addition space and contain all points whose coordinates are of the form $(3.5, \lambda, -2.0)$ where λ is any real number. In step 2 the intersection point I would therefore have as coordinates $(3.5, -5.5, -2.0)$. In step 3 the projection onto the range space R (which in this case corresponds to the y coordinate axis of the addition space) now yields a point with coordinate (-5.5) . This is indeed the result of the 'inverse addition' (subtraction) of 3.5 from -2.0 .

Similarly a choice of the yz as domain would have yielded the answer -5.5 (again the subtraction of 3.5 from -2.0) upon the final projection onto the x coordinate axis.

In each case using the geometric model of the addition plane to apply the addition function (or its inverse, the subtraction function) to a given pair of numeric values only made use of the basic modelling commands of the augmented polyhedral solid modelling system.

Another interpretation of the addition plane is as the graph of a mapping from a one-dimensional domain D into a two-dimensional range R . This case further contains several alternatives depending on whether the domain space of the mapping corresponds to the x , y or z coordinate axis of the addition space. The corresponding range spaces would then be the yz , xz and xy coordinate planes of the addition space. Obtaining the image of a point in the domain is independent of this choice as it involves following essentially the same sequence of steps as already described above.

For example, consider the point P_D in the one-dimensional domain space D with coordinate (2.5). Further assume that the space D corresponds to the x (or y or z) coordinate axis of the addition space. The two-dimensional range space R then corresponds to the yz (or xz or xy respectively) coordinate plane of the addition space. Applying step 1 of the above algorithm results in the geometric model X in the addition space. X is now a two-dimensional plane containing all points whose coordinates are of the form $(2.5, \lambda, \mu)$ (or $(\lambda, 2.5, \mu)$ or $(\lambda, \mu, 2.5)$ respectively) where λ and μ are any pair of real numbers. Step 2 of the algorithm produces the intersection I of X with the addition plane. I is now not a single point but a whole line containing all points with coordinates of the form $(2.5, \lambda, 2.5 + \lambda)$ (or $(\lambda, 2.5, \lambda + 2.5)$ or $(\lambda, 2.5 - \lambda, 2.5)$ respectively) where λ is any real number. The final projection of I onto the two-dimensional range space R in step 3 results not in a single point but in a whole geometric model consisting of the line of points whose coordinates are of the form $(\lambda, 2.5 + \lambda)$ (or $(\lambda, 2.5 + \lambda)$ or $(\lambda, 2.5 - \lambda)$ respectively) where λ is any real number (see Figure 5.3).

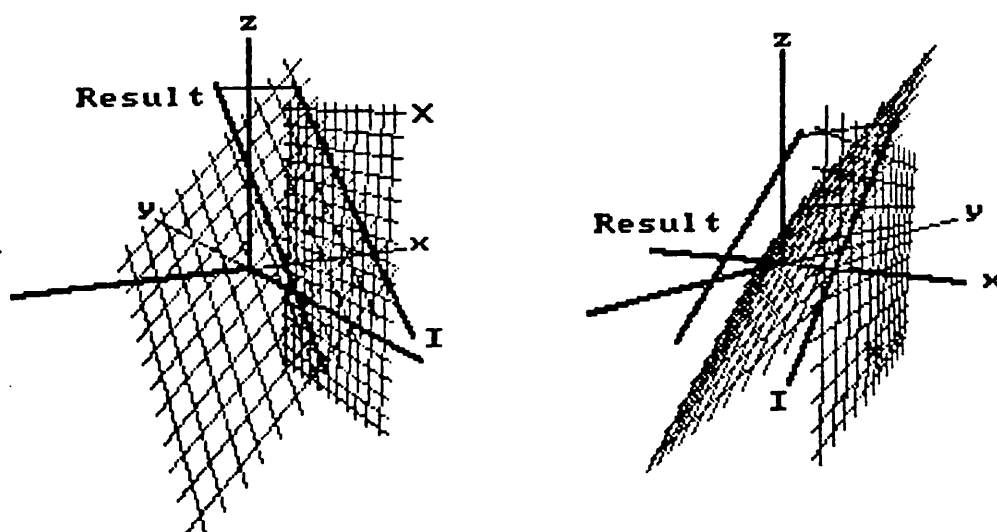


Figure 5.3. Applying the addition mapping.

The first two cases (where the domain D corresponded to the z and y coordinate axis respectively) can be interpreted as an example of partial evaluation of the addition function when only one of its two operands is supplied. The resulting geometric model is therefore (the graph of) the more 'specialised' function which takes only a single operand and adds 2.5 to it. On the other hand, the last

case (where the domain D corresponded to the z coordinate axis) effectively solved the question 'which pairs of numbers add up to 2.5' to find that in fact the answer is underconstrained and therefore non unique as all points in the resulting geometric model exhibit that property.

The mapping from a one-dimensional domain into a two-dimensional range is no longer a proper function because each point in the domain does not have, at most, one image point. Nevertheless, the image of a point under such a mapping can still be constructed by the same sequence of primitive modelling operations and be represented within the modelling system.

The next Section now considers the more complex case of a constructively defined geometric model.

5.2 THE MULTIPLICATION SURFACE

In this example the geometric model is a constructively defined infinite surface which happens to be the graph of the multiplication function $z = g(x, y) = x \cdot y$ as shown in Figure 5.4.

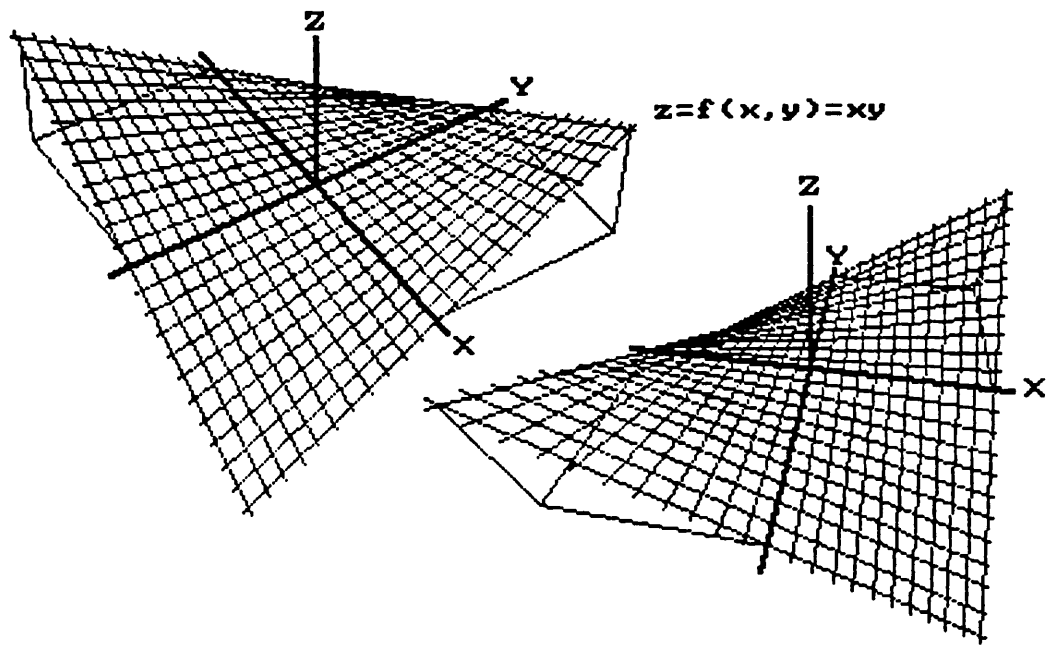


Figure 5.4. The multiplication surface.

The graph of the multiplication function $z = \text{Multiply}(x, y) = x \cdot y$ is a curved, ruled surface in a three-dimensional ‘multiplication’ space whose coordinate axes correspond to the x , y and z values.

This ‘multiplication’ surface cannot be represented directly in the proposed modelling because it is not a polytope. Instead it must be defined constructively by a bounding polytope and a sequence of modelling instructions which can be applied to obtain a set of more accurate (ie smaller) bounding polytopes.

As the multiplication surface extends infinitely, its bounding polytope must also be infinite. However, owing to the use of homogeneous coordinates it is difficult to represent it in one piece (see Section 4.2.1). Therefore the multiplication surface

is initially approximated by four bounding polytopes. Each of these is a tetrahedron which encloses one quarter of the multiplication surface. For example the first tetrahedron T_1 is defined as the Join (convex hull) of the points whose homogeneous coordinates are $(0,0,0,1)$, $(1,0,0,0)$, $(0,1,0,0)$ and $(0,0,1,0)$ — that is the origin O of the coordinate system of the multiplication space and the points at infinity X , Y and Z at the positive extremes of the x , y and z coordinate axes respectively (see Figure 5.5).

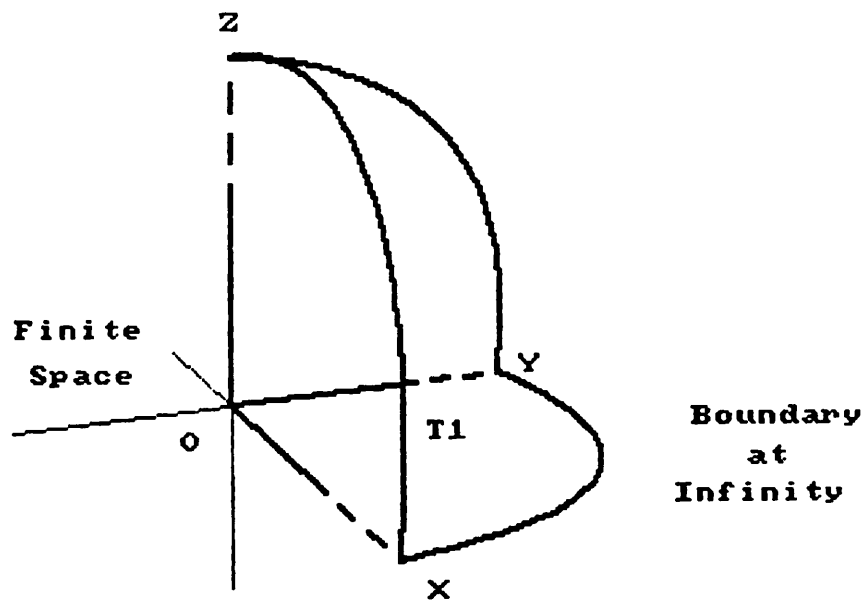


Figure 5.5. Bounding tetrahedron T_1 .

The second part of the constructive definition is the sequence of modelling operations which can be used to obtain more detailed bounding polytopes, typically consisting of a set of smaller polytopes similar to the initial one.

Consider the bounding polytope T_1 described above. The method of construction of a more accurate bounding envelope than T_1 consists in obtaining five new points $P_1, \dots, P_5$ which, together with the extreme points (vertices) O , X , Y and Z of T_1 , define four (smaller) tetrahedra T_{11} , T_{12} , T_{13} and T_{14} . Together these smaller tetrahedra form a more closely fitting envelope of the multiplication surface.

The proposed method of construction involves the construction of two intermediate lines L_{AB} and L_{CD} and proceeds as follows. Consider a (finite) point P of T_1 with (homogeneous) coordinates $(p_x, p_y, p_z, 1)$. P can be obtained from T_1 by considering T_1 not as a whole geometric entity but as a set of (infinitely many) individual points which can be enumerated systematically using the method outlined in Section 3.3.2. The projection of P onto the xy coordinate plane of the multiplication space will have homogeneous coordinates of the form $(p_x, p_y, 0, 1)$. Using only orthogonal projection and Cartesian product with the constants 0 and 1 four new points can be constructed. These are $A(p_x, 0, 0, 1)$ and $B(p_x, 1, p_x, 1)$ whose least enclosing linear space (ie linear hull) is the line L_{AB} and $C(0, p_y, 0, 1)$ and $D(1, p_y, p_y, 1)$ whose linear hull is the line L_{CD} (see Figure 5.6). Note that both lines L_{AB} and L_{CD} actually lie in the multiplication surface.

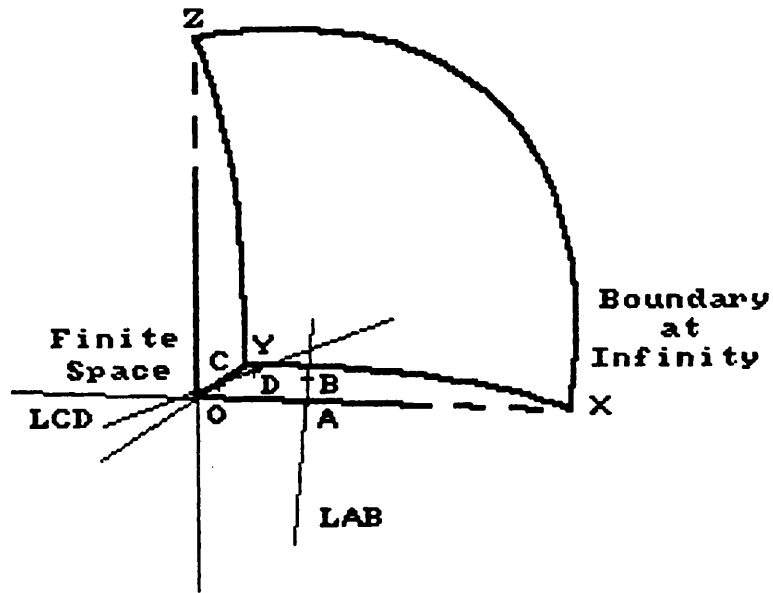


Figure 5.6. Constructive definition—Part 1.

Finally the required five new points are obtained as follows. P_1 and P_2 are the intersection points of the line L_{AB} with the original bounding tetrahedron T_1 (where P_1 denotes the point closest to the x coordinate axis of the multiplication space). Similarly, P_3 and P_4 are the intersection points of the line L_{CD} with the original bounding tetrahedron T_1 (where P_3 denotes the point closest to the

y coordinate axis of the multiplication space). The last point, P_5 , is simply the intersection of the two lines L_{AB} and L_{CD} (see Figure 5.7).

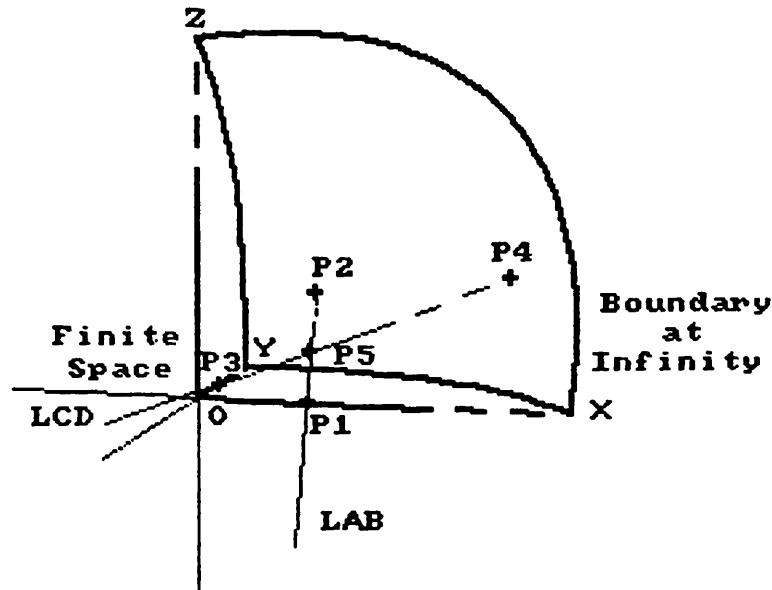


Figure 5.7. Constructive definition—Part 2.

The new, more accurate envelope of the multiplication surface is formed by the four tetrahedral polytopes T_{11} , the Join (convex hull) of the points O, P_1, P_3, P_5 , T_{12} the Join of P_1, X, P_5, P_4 , T_{13} the Join of P_3, P_5, Y, P_2 and T_{14} the Join of P_5, P_4, P_3, Z . These replace the original polytope T_1 which was the Join of O, X, Y, Z . Figure 5.8 shows both the result of the constructive definition and the effect of repeating it on the resulting smaller tetrahedra.

The methods of construction for the other three tetrahedra of the initial envelope of the multiplication surface are analogous to the one described above. However, it should be emphasised that the method of construction is, in general, not unique and that an alternative sequence of operations and indeed alternative enveloping polytopes could have been used.

Instead of being restricted to the interpretation of this model of the multiplication surface as a purely geometric model, it can also be used as a functional relationship in the same way as was illustrated above with the addition plane.

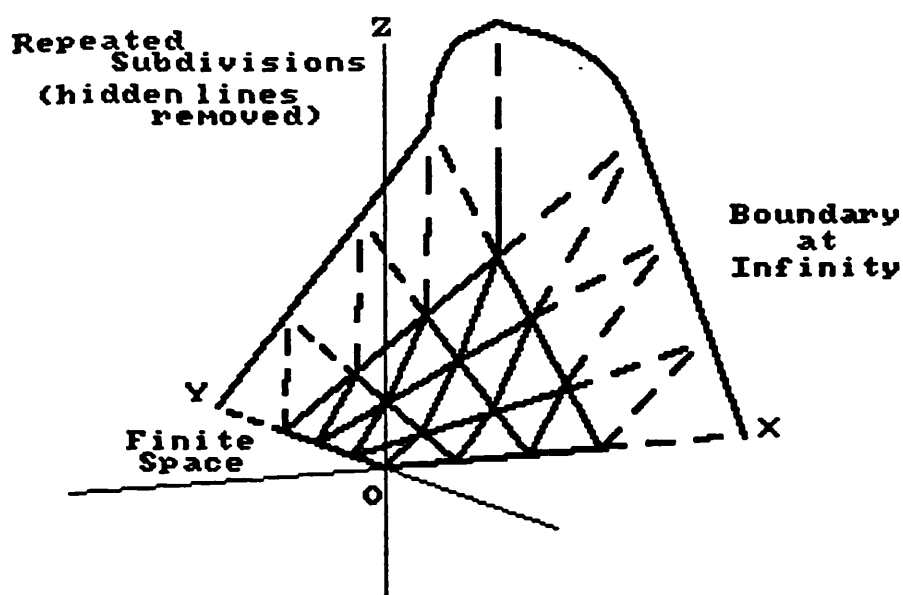
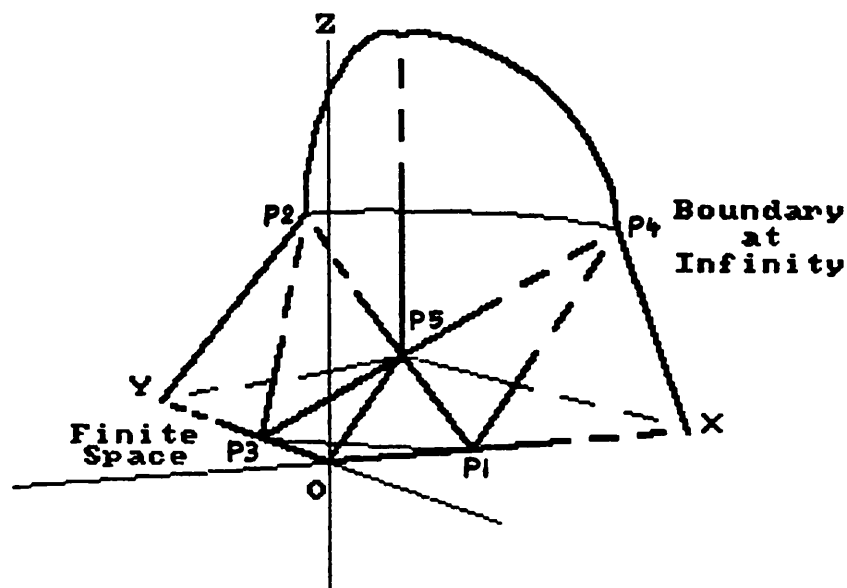


Figure 5.8. Constructive definition—Part 3.

The main difference lies in the additional complexity of the intersection computation which now involves the constructively defined model of the multiplication surface. Note that the intersection step always involves a well defined linear space L which is orthogonal to a given coordinate axis or plane of the multiplication space. The simplest solution consists of obtaining a constructively defined result

whose method of construction is the same as that of the multiplication surface with the intersection with the linear space appended as a final constructive operation. This effectively postpones any decision about 'how detailed' an answer should be constructed : the method of construction is altered, but no construction is actually performed. As the intersection entity will always be confined to the linear space L , a more elegant possibility would be to derive a method of construction which is completely contained within L ; that is one which only involves operations on entities of L within L . The automatic derivation of such a constructive definition is likely to be similar to the concept of automatic program transformation in functional programming languages and is an area which requires further research.

The various mappings that can be extracted from the multiplication surface $z = x \cdot y$ have the following interpretations :

$$\left\{ \begin{array}{ll} (x, y) \mapsto (z) & \text{multiplication } z = x \cdot y. \\ (x, z) \mapsto (y) & \text{division } y = z/x. \\ (y, z) \mapsto (x) & \text{division } x = z/y. \\ (x) \mapsto (y, z) & \text{line of points } (y, x \cdot y). \\ (y) \mapsto (x, z) & \text{line of points } (x, y \cdot x). \\ (z) \mapsto (x, y) & \text{hyperbolic curve } x \cdot y = z. \end{array} \right.$$

That the last case is indeed a hyperbola can be shown by performing a substitution using $X = \frac{y+x}{2b}$ and $Y = \frac{y-x}{2a}$ where a and b are such that $z = a^2 b^2$. This is equivalent to $x = bX - Ya$ and $y = bX + Ya$ so that $x \cdot y = z$ can be restated as $\frac{X^2}{a^2} - \frac{Y^2}{b^2} = 1$ which is indeed the definition of a hyperbola.

This Section together with its preceding one mainly illustrated the flexibility of the geometric knowledge representation system in relation to numerical mappings. The next Section briefly illustrates the use of the excursion box hierarchy for computing the intersection of two geometric models.

5.3 THE EXCURSION BOX HIERARCHY

This Section briefly illustrates how the excursion box hierarchy can be used to perform an intersection computation. For this purpose consider the two geometric models represented by their excursion boxes shown in figure 5.9.

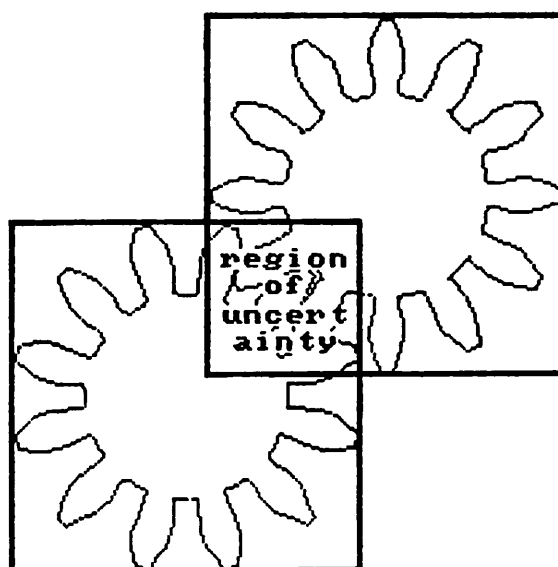


Figure 5.9. Two intersecting excursion boxes.

As the excursion boxes in Figure 5.9 overlap the intersection (or non-intersection) of the contained geometric models is subjected to the uncertainty prevailing in the region of overlap. As already suggested in Section 3.3.2 the size of the region of overlap could in some sense be used as a measure of this uncertainty. However, to obtain a less uncertain answer, the two original excursion boxes must be replaced by a more detailed representation. This essentially consists of two sets of smaller excursion boxes as shown in Figure 5.10.

How these sets of smaller excursion boxes are obtained depends on the way that the two initial excursion boxes were defined in the first place (see Section 3.3.2).

These two sets of excursion boxes can then be used to further reduce the area of uncertainty or at least identify more accurately which parts of the geometric

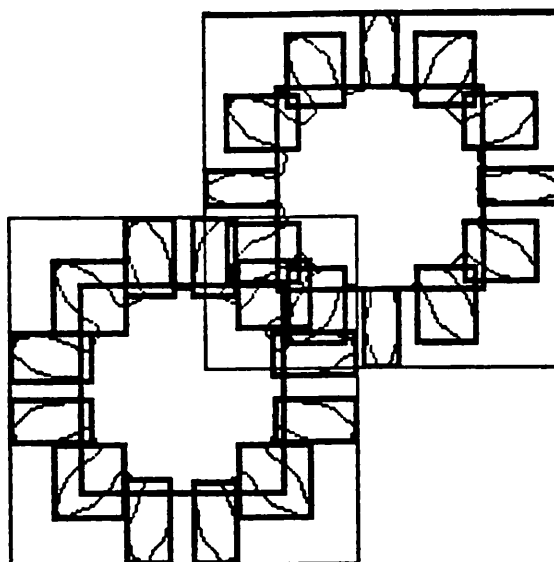


Figure 5.10. Refining the excursion boxes.

models are not, or may be, involved in the intersection. Comparing each of the smaller excursion boxes from one geometric model with each of the smaller excursion boxes from the other would be a simple but not necessarily very efficient option. A more efficient method is first to reduce the number of smaller excursion boxes describing each of the geometric models. This can be done by eliminating all excursion boxes which are clearly remote from the intersection, for example, all those which do not intersect the original region of overlap. In either case the original region of overlap will be replaced by one or more new, smaller regions of overlap as shown in Figure 5.11.

All the above comments now apply to each pair of overlapping smaller excursion boxes, and, if needed, these can be in turn replaced by sets of smaller, more accurate ones. This process may then be repeated as often as needed until one of the three possible termination conditions is reached.

The first possible termination condition is that the area of overlap has become small enough to be insignificant so that the two geometric models may be assumed to be simply in contact as shown in Figure 5.12.

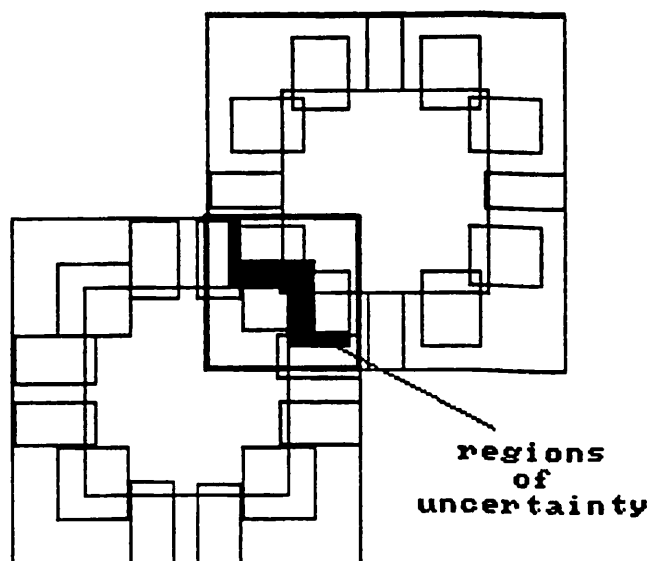


Figure 5.11. Refined excursion box overlap.

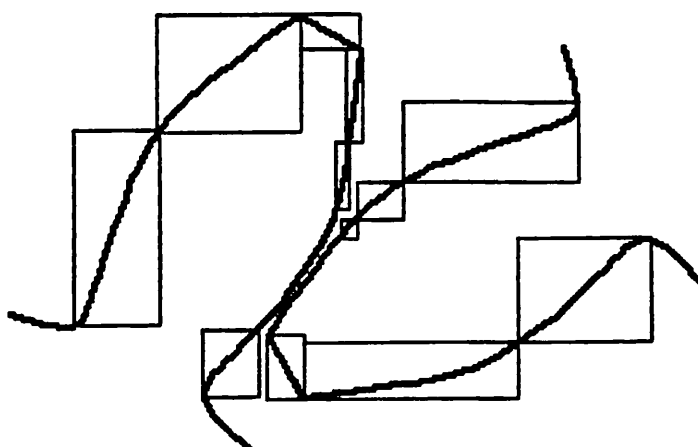


Figure 5.12. Two objects in contact.

The second possible termination condition is that at some level all the excursion boxes from the two geometric models are disjoint and there is no intersection. This case is shown in Figure 5.13.

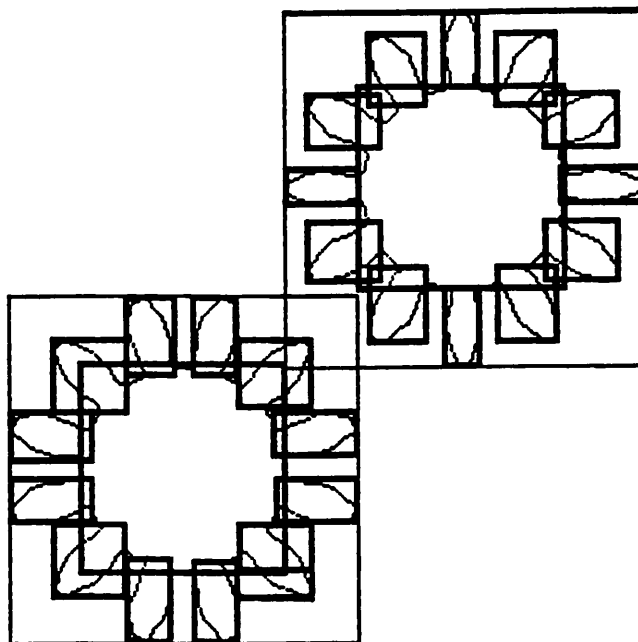


Figure 5.13. Two disjoint objects.

The final termination condition occurs when the level of excursion box refinement is pursued until two kinds of excursion boxes emerge : those straddling the boundary of the geometric model and those completely contained within its interior (see Section 3.3.2). The termination condition arises when two 'interior' excursion boxes overlap. In this case the two geometric models do clearly intersect as shown in Figure 5.14.

It follows from the above description that the process of 'refining' a pair of excursion boxes into two sets of smaller, more accurate excursion boxes is subjected to two opposing tendencies as a large number of smaller excursion boxes will on the one hand lead to a rapid reduction in the area of uncertainty but on the other increase the amount of computations needed to do so.

This method of intersection computation may or may not be an optimal method depending on the nature of the geometric models involved. The essential feature which makes it so valuable is that it will work with any pair of geometric models independently of their complexity of definition. In particular it can be used to determine the equality of two constructively defined models to within any arbitrary level of resolution (as measured by the size of the resulting region

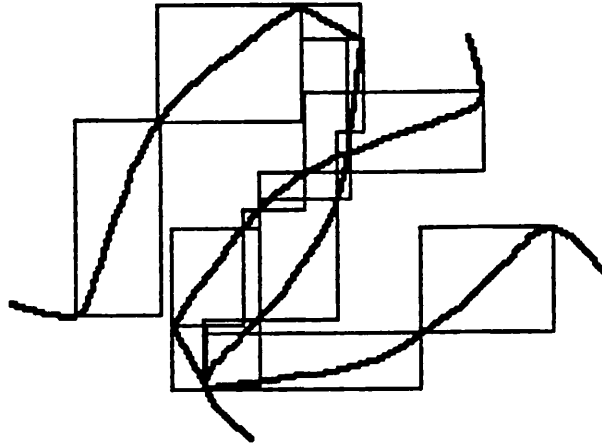


Figure 5.14. Two intersecting objects.

of overlap), even though this may become increasingly more computationally expensive at finer levels of resolution.

The above example illustrates the intersection computation at the level of two individual components. However, the benefit of this method mainly lies in the case where two complex assemblies each comprising many components (see Section 3.3.2) must be intersected. It is in this case that the elimination of the remote excursion boxes during each successive refinement will in fact discard whole sub-assemblies and groups of components and thus allow the algorithm to home in rapidly on the relevant ones. This therefore avoids the laborious testing of each individual component in one assembly against all individual components in the other.

The example in this Section mainly illustrated the use of the excursion box hierarchy for computing the intersection of two geometric models. However, the example in next Section will take place in a context more related to artificial reasoning.

5.4 POSSIBLE GEOMETRIC REASONING

One of the features of the proposed system of representation is to enable information best modelled by continuously varying parameters to be represented as a finite set of discrete 'qualitatively' distinct regions. In this context, the purpose of a geometric "deduction" would be to construct such finite sets which corresponded in some way to a particular property. The following, very simple two-dimensional example will try to illustrate how this could be achieved.

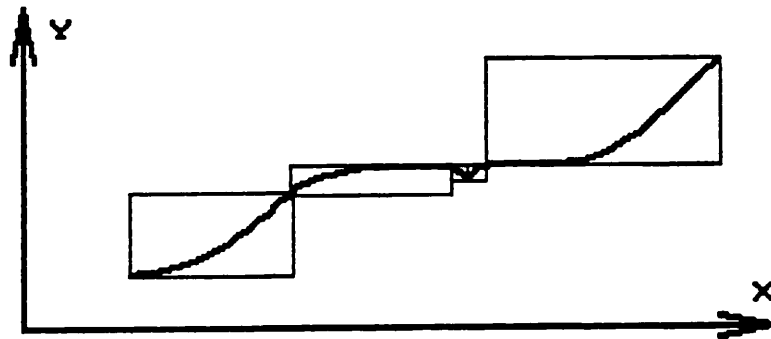


Figure 5.15. Initial geometric model G .

Consider the geometric model G shown in Figure 5.15 together with its excursion box description. This is represented in the computer as a list of geometric models, some straight and some curved (and therefore defined constructively). This model may be interpreted as part of the outline of some engineering component.

Using a constructive finite difference method the slope at each point of the above geometric model G can be both defined and calculated. The resulting geometric model of the slope of G , together with its excursion boxes is shown in Figure 5.16.

Using the excursion box decomposition method this new geometric model can be described in greater detail, for example by ensuring that all excursion boxes are less than some given size. Figure 5.17 illustrates the level of description that is obtained when all excursion boxes are decomposed until they are narrower than a given reference value R along the x coordinate axis.

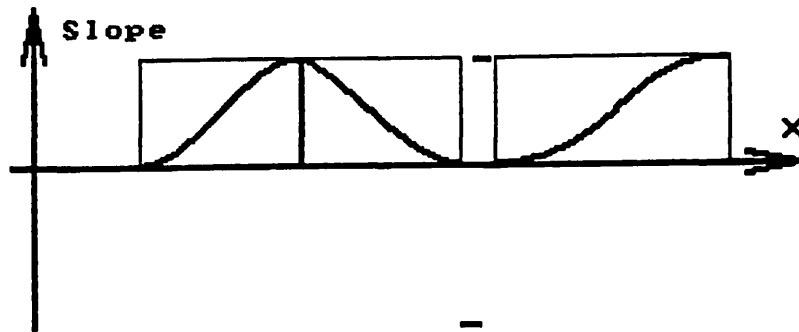


Figure 5.16. Geometric model of the slope of G .

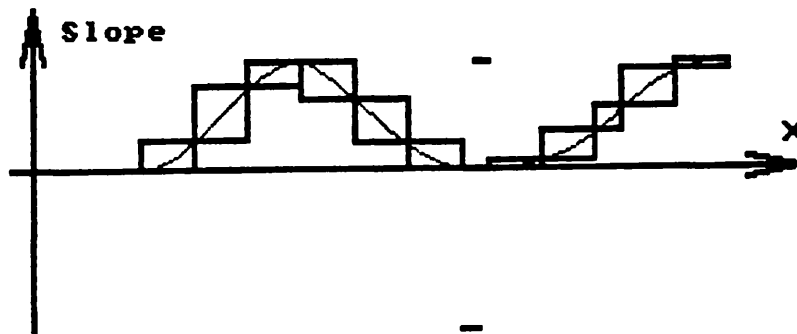


Figure 5.17. Detailed model of the slope of G .

Each of these small excursion boxes and each adjacent pair of excursion boxes can be compared to a reference box B_R whose extent along the x coordinate axis was equal to R . Any excursion box which is "taller" than B_R , or any pair of adjacent boxes which are separated by more than the height of B_R can be found and labelled. Adjacent excursion boxes with same label can be grouped together to form larger excursion boxes as shown in Figure 5.18.

Finally, the set of larger excursion boxes which resulted from the last operation can be mapped back onto the original geometric model by reversing the initial constructive procedure. This results in a set of excursion boxes on G as shown in Figure 5.19.

The interpretation of this "geometric deduction process" is that it locates those regions of G where the slope of the tangent to G changes by more than Δ_{Slope}

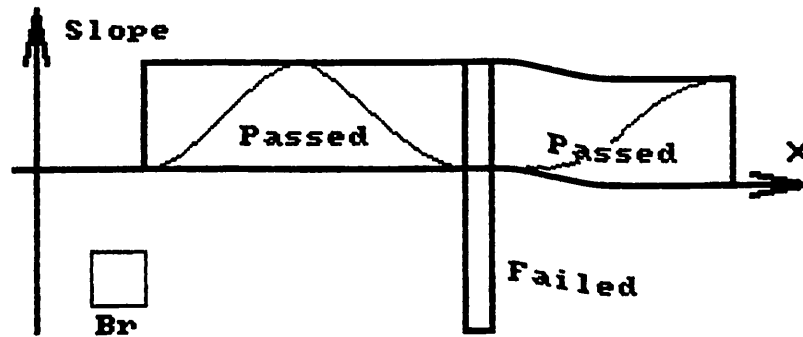


Figure 5.18. After the comparison with B_R .

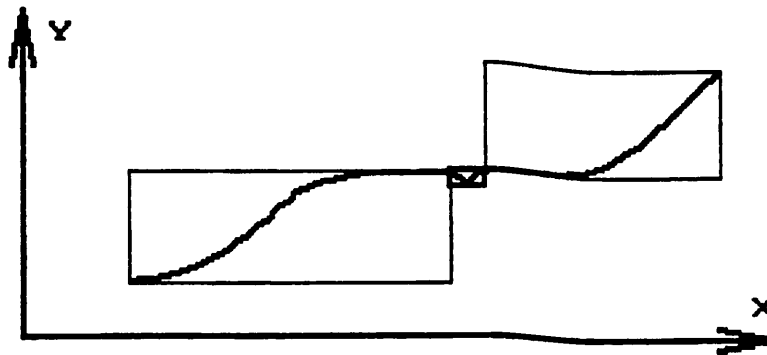


Figure 5.19. A new partition of G .

over an interval Δ_x , where Δ_{Slope} and Δ_x are respectively the height and width of the reference excursion box B_R . The resulting set of excursion boxes identifies those regions of the original geometric model where there is a change of slope which is greater than a specified threshold, a feature which is of importance in a stress analysis context.

The proposed method of deduction illustrated in the above example is as follows. The starting point is a geometric model in some space S_1 . This is used as the “domain” for the constructive definition of a second geometric model in some other space S_2 . This second geometric “image” model is then tested using some simple excursion box based criterion within S_2 . The contiguous regions of similar response are then grouped (and enclosed in excursion boxes). The resulting distinctions on the “image” model in S_2 are then mapped back onto the original geometric “domain” model in S_1 . The set of excursion boxes which results from

this last operation distinguishes regions of the original model which correspond to a particular state of the property that was tested for in the (property) space S_2 .

The critical step of this potential deduction process occurs when the excursion boxes have to be found in S_2 for the constructively defined images of the individual parts of the domain model in S_1 . The problem is that of determining the upper and lower bounds of the constructively defined geometric entities purely from the description of the “domain” entities and the particular method of construction. How this computation may be achieved and under what conditions it is, or is not, possible is an open question which requires further research. The qualitative deduction methods reviewed in Section 2.4 might form a possible starting point for such an investigation.

A comparison of the proposed deduction system with existing deduction systems such as the logic programming language Prolog is difficult. For instance, the deduction system of Prolog uses many ‘rules’ to enumerate the specific cases in which some conclusion holds. On the other hand, the proposed deduction system manipulates geometric models and provides its answer in terms of geometric models. However, using the equivalences suggested in Section 3.1, any knowledge expressed in Horn Clauses can be represented in the proposed modelling system (through the addition of suitable coordinates) and conversely a geometric model can be represented within Prolog or any other general purpose programming language. At this level the advantage of the proposed system reduces to the provision of primitives which are better aligned with the goals of the user in a CAD context. This is analogous to the advantage of a ‘high level’ computer language over the direct entry of hexadecimal machine operation codes for the purpose of developing computer programs.

This completes the “geometric deduction” example. The purpose of this and the previous examples is finally summarised in the following Section.

5.5 EXAMPLES SUMMARY

The example of the addition plane illustrated the definition of an ‘infinite’ geometric model and the multiplication surface provided an example of a geometric model defined indirectly by a method of construction.

The interpretation of a purely geometric model as a piece of geometric knowledge was illustrated by alternately considering each of the above examples as the graph of a mapping or functional relationship. Each possible choice of mapping was further shown to represent a meaningful interpretation of the geometric model.

Although many of the interpretations corresponded to the basic arithmetic operations of addition, subtraction, multiplication and division, the basic unit of representation remained the geometric model rather than the real number. The link between the two is that a number can be represented by (the single coordinate of) a point in a one-dimensional space and is therefore a special case of a geometric model. The concept of geometric model is, however, more powerful as was illustrated above by the cases which mapped a point from a one-dimensional domain space into a whole geometric model in a two-dimensional range space.

Taken together the examples of the addition plane and multiplication surface prove that through a suitable concatenation of the constructive steps constituting the ‘mapping application’ algorithm any finite, arbitrary arithmetic expression can be evaluated. Therefore the proposed formalism for the representation of geometric knowledge is, in theory, sufficiently powerful to perform numerical computations.

The use of a piece of geometric knowledge as a geometric model was also briefly illustrated indirectly when, during step 2 of the ‘mapping application’ algorithm, the intersection between two geometric models was computed. The use of the excursion box hierarchy for performing or assisting in the geometric modelling operation of intersection was illustrated in the example of Section 5.3.

The final example (Section 5.4) illustrated how the proposed system for the representation of geometric knowledge might be able to support deduction processes on the geometric models.

Taken together the examples in Sections 5.1 to 5.4 demonstrate the flexibility of the proposed scheme of geometric representation of knowledge for CAD as it combines the expressive power of arithmetic (Sections 5.1 and 5.2) and geometric modelling (Section 5.3), and offers the possibility of a system of deduction (Section 5.4).

Note that the purpose of automating deduction activities which are relatively simple to perform for human beings is motivated by the the desire to complement human creativity with the systematic consistency of computerised knowledge. For example, the relationships governing the physical world, once formulated should always be born in mind during a design process. A possible application area would therefore be the automatic integrity verification of a proposed design, a task which becomes increasingly important with larger, more complex designs. Such a system could relieve the designer from managing *all* the details of such information consistently *all* of the time (by providing a limited kind of 'safety net') and therefore free him to concentrate more on, for example, value engineering aspects of his design.

With this last idea the argumentation of this Thesis has completed a full circle and returned to the initial statements of the Chapter 1. The next, and final, Chapter will therefore provide an overall summary and conclusion.

Chapter 6. CONCLUSIONS AND FUTURE WORK

6.1 CONCLUSIONS

The main result of this research has been to propose a powerful and flexible representation scheme for geometric information or knowledge in a Computer Aided Design context.

The Thesis began by identifying a representational need in Artificial Intelligence with respect to a type of information frequently required in mechanical engineering design. The information in question is the representation of, and interaction with, geometric shapes of solid objects. The fundamental nature of the difficulty was traced to the use of *continuously varying* parameters to represent such information within the context of deduction systems operating on *discrete* representations. The rest of the Thesis then systematically developed and refined a method of overcoming this problem.

The first step extended and generalised the statement of the problem from the restricted context of the representation of the shape of solid objects to any information which is best represented by means of continuously varying parameters (such as geometric entities) in a real number space. In particular, this includes any continuous relationship linking engineering quantities.

Two existing schools of thought were used to assist with this step. The first was the Horn Clause subset of Predicate Logic which demonstrated a representational requirement for multiple spaces of any finite dimension (as analogues of multiple predicates of any finite arity). The second was Qualitative Reasoning which in turn demonstrated the necessity of being able to represent unbounded quantities as well as finite ones.

A theoretic justification for the correctness of such representational requirements was sought and found in the form of a pair of Theorems in Join Geometry (a non-dimensional approach to convex set theory) and point set Topology. This led to the postulation of a generalised polyhedral modeller as the means of representing information involving continuously varying parameters.

At this stage an additional hierarchical representational element was introduced by borrowing the concept of excursion boxes. These provide a systematic way of representing geometric uncertainty or approximate (geometric) information and were shown to be a natural generalisation from the one-dimensional system of representation of Qualitative Reasoning.

The notion of constructive definitions was next introduced as a means of extending the modelling capabilities and acting as a form of computation similar to the concept of lazy evaluation in functional programming languages.

At this stage the central idea was presented for elevating a geometric model to the (geometric) representation of a piece of knowledge which involves a continuously varying parameter. This simply consisted of a theoretical framework for enumerating all possible modes of interpretation or use of such a geometric model.

Finally, the augmented requirements on a qualitative representation scheme were briefly considered. These were found to originate from three sources, namely the generalisation to higher dimensions, the generalisation to mappings and relationships which are not mathematical functions and the introduction of a hierarchy.

The emphasis of the Thesis then shifted from theoretic considerations to the more practical issues surrounding the implementation of the proposed system. The first half of this process discussed the kinds of data structures that would be capable of dealing with such new representational requirements as points at infinity and convex polyhedra (polytopes) in higher dimensions. The second, was the detailed description of the principal (geometric) modelling algorithms most of which were also implemented.

Finally some examples were used to illustrate the flexibility of the proposed system of geometric knowledge representation which can combine the expressiveness of arithmetic and geometric modelling into a coherent whole with the prospect of supporting automatic deduction processes in the future.

6.2 FUTURE WORK

Throughout the main body of the Thesis, areas which required further research were clearly identified.

In particular, in the short term the two main directions of further work are enlarging the implementation and resolving some of the open theoretical issues. The former involves completing the modelling algorithms and incorporating the constructive definitions and excursion box hierarchy. The latter involves three tasks : formalising the concepts of constructive definitions as a first step towards studying their strengths and limitations; formalising the extensions to qualitative representation schemes to include the effects of higher dimensions, non-functional relationships, and a hierarchical structure. Together these steps should clarify the feasibility of the outlined deduction system.

In the longer term the emphasis of the research would shift towards exploring the type of higher level structures that would be required to use the proposed system to express more complex 'engineering' information and thus be in a position to address the information manipulation issues raised by design theory and suggested by the various design methodologies.

The final Section considers some of the implications of the ideas presented within this Thesis.

6.3 CONSEQUENCES

This Section is on a much more philosophical and speculative note as it considers some of the consequences of the ideas developed within this Thesis. These are presented in the following Sub-Sections.

6.3.1 Representation of Uncertainty

The first idea is associated with the concept of uncertainty. It could be argued that, in the real world, information at any level in a hierarchy of 'amount of detail' will always have some uncertainty associated with it. Therefore, in the real world there would appear not to be a base of 'absolute certainty' to any information no matter how detailed, but only an undecomposable entity at the frontier of knowledge — the 'beyond' is simply unknown. For example, in the case of the model of a solid object the assumption of a continuous surface breaks down at a very small scale and should be replaced by alternative models which reflect the atomic structure and, at a still smaller scale, the fundamental particles whose structure (if any) remains unknown.

The hierarchical structure of knowledge which emerges is that a given model only represents some knowledge over a certain range of uncertainty before it has to be replaced by another. The 'base' of this hierarchy is undecomposable and represents the minimum amount of uncertainty beyond which it is not possible to progress. Such information at the boundary of knowledge could be represented within the proposed system simply by means of excursion boxes which can not be decomposed any further.

The benefit from the design point of view is that this mechanism would be able to represent the kind of partial information that exists during a design process before the design is completed. The meaning of such excursion boxes would be to serve as a rough approximation to the region of space that is being considered. If the space is three-dimensional physical space, then the excursion box might be a guide to the location, size and shape of a proposed component which will be contained in the excursion box; if on the other hand the space represents some performance characteristic, the excursion box may be used as a guide to the area of expected performance, or more simply as a region of desired (or forbidden) operating conditions. The benefit is that even though the contents of such excursion boxes has not yet been determined, they nevertheless provide an approximation to their future content and thus allow some decisions to be made with essentially uncertain or vague information.

6.3.2 Modelling Aspects of Common Sense

The second idea follows on from the first and considers its implications on what is best termed 'common sense'.

One aspect of 'common sense' relates to the management of the amount of uncertainty present in any information. This applies in particular to design processes and is illustrated as follows. At the outset of a design project the uncertainty about the final product, and in particular the existence of components, or whole groups of components, may be more or less total. In this context it would be regarded as a violation of 'common sense' to produce very detailed drawings of all 'nuts and bolts' in a preliminary design sketch. The reason is that drawing the details of nuts and bolts means operating a level of detail which is far smaller than the uncertainty associated with the major constituents of the design — doing so is therefore not justifiable at this early stage of the design process.

The proposed representation system could be used to assist the control of the design process by keeping track of the distribution of uncertainty throughout the partial design which exists before the design process is completed. Such information would also be of value to a knowledge based design expert system in deciding what to do next; for example, it could advance those aspects of the design which corresponded to the largest current regions of uncertainty.

6.3.3 Representing Aspects of Design Experience

The third idea again follows on from the previous ones and suggests a means of representing an aspect of 'design experience'. It is introduced by means of the following example.

The mechanism which determines the level of uncertainty (or detail) which is acceptable (or necessary) in any given circumstance is exceedingly important. For example, when friction occurs the electrostatic charges created on the contacting entities are usually of no concern in a mechanical engineering context — except when large quantities of a flammable liquid are being transferred from one container to another. In this context the 'electronic' scale phenomena of static electricity assumes a very important role in the prevention of unwanted fires caused by static arcing. It could be argued that this phenomenon is not immediately obvious, but once attention is drawn to it (possibly as the result of an accident) it is sufficiently important to be incorporated into design knowledge in the form of 'experience'. In this case phenomena which can only be explained at a very much smaller scale than that needed for design purposes do have important consequences both on the final design and its operating conditions and procedures.

Therefore representing of the range of applicability of any particular model forms an important part of engineering 'experience' and 'common sense'. In the proposed system this kind of information could be expressed systematically in terms of a measure such as the size of the excursion boxes involved. As the size of the excursion boxes dropped below a certain threshold, the constructive procedure associated with the contained geometric model could be replaced by a new one which reflected the change in characteristics encountered at the smaller scale. An even better system would use two thresholds, one for the introduction of the new model and a second, later threshold for the removal of the old model. This would yield a range of scales in which both models (and any deductions made from them) had to be considered in parallel.

6.3.4 Self Representation

The examples in Chapter 5 suggest that the proposed system has the representational power of arithmetic. Furthermore, the constructive definitions allow sequences of modelling operations to be used as definitions of more complex models. Therefore it should be possible to represent all the modelling algorithms as models within the system itself. Similarly, the interactions of a user with the system also form a sequence of modelling operations. Therefore the user actions could also be represented directly within the system as yet another (probably very high dimensional) model. This would not only keep a record of the design process (as advocated by Design Theory — see Section 2.1) but also make it amenable to the full spectrum of operations supported by the system. In particular it would include replaying the design actions, modifying them or even obtaining deductions from them via geometric reasoning. This would provide a uniform format for keeping a record of both the design process and its results.

6.3.5 Extension to Other Knowledge

The final idea is that the advantages of the structure (and of the hierarchy in particular) inherent in the proposed system could be extended to information which is, by nature, not based on continuously varying quantities. This would involve placing any new fact (modelled as a point with symbols in lieu of coordinates) in a continuous metric space by providing it with a set of numeric coordinates. This process is effectively independent of the subject matter of the fact. The coordinates may therefore be chosen solely for the convenience of storage, for example in a hierarchical structure chosen so as to enhance the rapid access to that information (using the current pattern of usage as a guide).

In this case ‘spatial proximity’ could then be used as a second mechanism for addressing (this kind of) information in addition to purely logical relationships. Accessing information using spatial proximity would represent a non logical ‘though jump’ which would clearly be subject to logical validation to confirm that the result made any sense. Using such a mechanism for accessing information assumes that if related concepts are stored ‘near’ each other, then concepts stored

near each other will also be likely to be relevant to each other. In symbolic form, if B is stored 'near' A and C is stored 'near' A , then if B is also fairly 'near' to C that might imply a meaningful link between B and C . The assumption is that this case occurs sufficiently frequently to make this kind of 'information accessing' worth while. The term 'near' could be interpreted as either less than a given value of (a Euclidean metric of) distance or more simply (in terms of excursion boxes) if the concerned excursion boxes touch or overlap.

This Thesis having introduced the idea of representing information based on continuously varying quantities also forms a contribution to the area of knowledge representation in general. As illustrated by the ideas introduced in this last Section, this opens up new avenues of research which are likely to have implications that reach well beyond the field of CAD.

REFERENCES

- ABBO78 W.Abbot, T.H.Hewitt, "Technical Drawing (4th Edition)," Blakie and Son Limited, 1978.
- ADDA85 S.Addanki, E.Davis, "A Representation for Complex Physical Domains," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 443–446, 1985.
- ALSI86a J.Alsina, J.P.Fielding, A.J.Morris, "An Expert System for Aircraft Design," Technical Report 654–04 of the College of Aeronautics, Cranfield Institute of Technology, Bedford MK43 OAL, UK, 1986.
- ALSI86b J.Alsina, J.P.Fielding, A.J.Morris, "ADROIT : An Aircraft Design Aid," Technical Report of the College of Aeronautics, Cranfield Institute of Technology, Bedford MK43 OAL, UK, 1986.
- ANDR85 N.M.Andreasen, "Methodology for Layout Design ('Entwerfen')," in *Proceedings of the International Conference on Engineering Design ICED 85 : Theory and Practice of Engineering Design in International Comparison*, V.Hubka(Ed), Edition Heuristo (Switzerland), 257–267 (Vol.1), August 1985.
- AQUE83a A.Aquesbi, J.V.Boquet, J.M.Fouet, M.Reynier, et al., "An Expert System for Computer Aided Mechanical Design," in *Information Processing 83*, R.E.A.Mason(Ed), North Holland, 121–125, 1983.
- AQUE83b A.Aquesbi, "Application de L'Intelligence Artificielle a la Conception Mechanique : Le Systeme COPART," These du Diplome de Docteur de 3eme Cycle, Universite Pierre et Marie Curie (Paris 6, France), December 1983.
- ARCH85 B.Archer, "The Implications for the Study of Design Methods of Recent Developments in Neighbouring Disciplines," in *Proceedings of the International Conference on Engineering Design ICED 85 : Theory and Practice of Engineering Design in International Comparison*, V.Hubka(Ed), Edition Heuristo (Switzerland), 833–840 (Vol.2), August 1985.

- BAIL85 R.Bailey, "An Introduction to HOPE," Dept. of Computing, Imperial College, London SW7 2BZ, UK, (private communication), 1985.
- BALL81 D.H.Ballard, "Strip Trees: A Hierarchical Representation for Curves," *Communications of the ACM*, v24, n5, 310-321, May 1981.
- BATA83 J.Batali, "Dependency Maintenance in the Design Process," in *Proceedings of the IEEE International Conference on Computer Design : VLSI in Computers (ICCD'83)*, 459-462, November 1983.
- BELL86 J.Bell, M.Machover, "A Course in Mathematical Logic," North Holland, 1986.
- BERG66 C.Berge, "Espaces Topologiques : Fonctions Multivoques," Dunod, Paris, 1966.
- BERG63 C.Berge, "Topological Spaces," Oliver and Boyd, 1963.
- BOHA85 I.Bohachevsky, T.Cotter, V.Viswanathan, "Development of an Autonomous Design Program," DE 86002417/GAR PC A02/MFA01, Los Alamos Nat. Lab., 1985.
- BRAI86 I.C.Braid, "From Geometric to Product Modelling," in *Software for Discrete Manufacturing*, J.P.Crestin, J.F.McWaters(Ed), North Holland — IFIP, 123-133, 1986.
- BROW86 D.C.Brown, B.Chandrasekaran, "Knowledge Based Control for a Mechanical Design Expert System," *IEEE Computer*, v19, n7, 92-100, July 1986.
- BROW85a D.C.Brown, "Capturing Mechanical Design Knowledge," *Computers In Mechanical Engineering*, (private communication), 1985.
- BROW85b D.C.Brown, B.Chandrasekaran, "Plan Selection in Design Problem Solving," *AISB 85 Conference*, Warwick, UK, April 1985.
- BROW85c D.C.Brown, "Failure Handling in a Design Expert System," *Computer Aided Design*, v17, n9, 436-442, November 1985.
- CARL85 I.B.Carlbohm, I.Chakravarty, D.Vanderschel, "A Hierarchical Data Structure for Representing the Spatial Decomposition of 3D Objects," *IEEE Computer Graphics and Applications*, 24-31, April 1985.
- CASA87 M.S.Casale, "Free Form Solid Modelling with Trimmed Surface Patches," *IEEE Computer Graphics and Applications*, v2, n1, 33-43, January 1987.
- CASA85 M.Casale, E.L.Stanton, "An Overview of Analytic Solid Modelling," *IEEE Computer Graphics and Applications*, 45-56, February 1985.

- CHAN85 B.Chandrasekaran, R.Milne, "Special Section on Reasoning about Structure, Behaviour and Function," *Sigart Newsletter*, n83, 4-32, July 1985.
- CHAN70 D.R.Chand, S.S.Kapur, "An Algorithm for Convex Polytopes," *Journal of the Association for Computing Machinery*, v17, n1, 78-86, January 1970.
- CHOL83a L.Cholvy, J.Foisseau, "ROSALIE : a CAD Object Oriented and Rule Based System," in *Information Processing 83*, R.E.Mason(Ed), North Holland, 501-505, 1983.
- CHOL83b L.Cholvy, "Structuration et Integrite des Information dans les Bases de Donnees en C.A.O.: definition d'un modele de donnees et realisation d'une maquette," These de Docteur-Ingenieur, Ecole Nationale Supérieure de l'Aeronautique et de l'Espace, Toulouse, France, 1983.
- CLOC81 W.F.Clocksinn, C.S.Mellish, "Programming in Prolog," Springer Verlag, 1981.
- CULL86 S.J.Culley, J.Vogwell, S.W.Lye, "Design Decision Making Using Component Databases," Design Group, School of Engineering, University of Bath, UK, (private communication), 1986.
- DARL85 J.Darlington, A.J.Field, H.Pull, "The Unification of Functional and Logic Languages," Department of Computing, Imperial College of Science and Technology, London, (private communication), 1985.
- DARL81 J.Darlington, "The Structured Description of Algorithm Derivations," in *Algorithmic Languages*, DeBakker, Van Vliet(Ed), IFIP — North Holland, 221-250, 1981.
- DAVI83 E.Davis, "The Mercator Representation of Spatial Knowledge," in *Proceedings International Joint Conference on Artificial Intelligence*, 295-301 (Vol.1), 1983.
- DAVI77 M.Davis, "Applied Non-Standard Analysis," J.Wiley and Sons : Pure and Applied Mathematics — Intersciences Series, 1977.
- DEKL84 J.DeKleer, J.S.Brown, "A Qualitative Physics Based on Confluences," *Artificial Intelligence*, v24, 7-83, 1984.
- DIXO87 J.R.Dixon, et al, "Expert Systems for Mechanical Design : Examples of Symbolic Representations of Design Geometries," *Engineering with Computers*, v2, n1, 1-10, 1987.

- DIXO85 J.R.Dixon, E.L.Libardi, J.S.C.Luby, M.Vaghul, M.S.Simmons, "Expert Systems for Mechanical Design : Examples of Symbolic Representations of Design Geometries," in *Applications of Knowledge Based Systems to Engineering Analysis and Design: Proceedings of the Winter Annual Conference of the ASME*, ASME — AD-10, C.L.Dym(Ed), 29-46, 1985.
- DYER86 M.G.Dyer, M.Flowers, J.Hodges, "EDISON : An Engineering Design Invention System Operating Naively," *International Journal of Artificial Intelligence in Engineering (GB)*, v1, n1, 36-44, July 1986.
- DYM_85 C.L.Dym, "Expert Systems : New Approaches to Computer Aided Engineering," *Engineering with Computers*, v1, n1, 9-25, 1985.
- EAST78 C.M.Eastman, "The Representation of Design Problems and Maintenance of their Structure," in *Artificial Intelligence and Pattern Recognition in Computer Aided Design — Proceedings of IFIP WG5.2*, J.C.Latombe(Ed), North Holland, 335-357, 1978.
- EDEL81 H.Edelsbrunner, H.A.Maurer, "On the Intersection of Orthogonal Objects," *Information Processing Letters*, v13, n4,5, 177-181, End 1981.
- EGGL69 H.G.Eggleston, "Convexity," Cambridge University Press, 1969.
- FINK83 L.Finkelstein, A.C.W.Finkelstein, "Review of Design Methodology," *IEE Proceedings Part A*, v130, n4, 213-221, June 1983.
- FOLE82 J.Foley, A.Van Dam, "Fundamentals of Interactive Computer Graphics," Addison Wesley — Systems Programming Series, 1982.
- FORB84 K.Forbus, "Qualitative Process Theory," *Artificial Intelligence*, v24, 85-168, 1984.
- GAME83 T.W.Gamelin, R.E.Greene, "Introduction to Topology," Saunders College Publishing, 1983.
- GRUN67 B.Grunbaum, "Convex Polytopes," J.Wiley and Sons — Interscience Publishers, 1967.
- GUER86 E.Guerriery, V.Grover, "Octree Solid Modelling with Prolog," in *Application of Artificial Intelligence to Engineering Problems*, D.Sriram, R.Adey(Ed), Springer Verlag, 985-1001 (Vol.2), 1986.
- GUIL85 J.Guillot, J.Y.Rousselot, J.C.Vignat, "SICAM : une Approche Modulaire de la Conception de Mecanismes," in *Proceedings Of MICAD 1985*, 124-137, 1985.

- HAMM87 P.Hammond, M.Sergot, "Augmented Prolog For Expert Systems (APES) System Documentation 3rd Edition," Logic Based Systems Ltd, 1987.
- HOWE86 A.Howe, P.Cohen, J.Dixon, M.Simmons, "Dominic : A Domain Independent Program for Mechanical Engineering Design," in *Applications of Artificial Intelligence in Engineering Problems — Proceedings of the 1st International Conference*, D.Sriram, R.Adey(Ed), Springer Verlag, 289–299 (Vol.1), April 1986.
- HUGH85 J.Hughes, "Why Functional Programming Matters," Programming Methodology Group Memo PMG-40, Institutionen for Informationsbehandling, Chalmers Tekniska Hogskola (41296 Goteborg, Sweden), 1–20, 1985.
- JAKO85a K.Jakobsen, "Transition between Product Models," in *Proceedings of the International Conference on Engineering Design ICED 85 : Theory and Practice of Engineering Design in International Comparison*, V.Hubka(Ed), Edition Heuristo (Switzerland), 198–205 (Vol.1), August 1985.
- JAKO86b D.E.Jakopac, "A Multi-Level, Model Baseed Planner for Mechanical Assembly," PhD Thesis, Northwestern University, August 1985.
- JOHN85 T.John, K.Ehrlenspiel, "Methode zur Funktionellen Systembeschreibung," in *Proceedings of the International Conference on Engineering Design ICED 85 : Theory and Practice of Engineering Design in International Comparison*, V.Hubka(Ed), Edition Heuristo (Switzerland), 206–212 (Vol.1), August 1985.
- KELL55 J.L.Kelly, "General Topology," Springer Verlag, 1955.
- KITA84 K.Kitajima, H.Yoshikawa, "HIMADES-1 : a Hierarchichal Machine Design System Based on the Structure Model for a Machine," *Computer Aided Design*, v16, n6, 279–307, November 1984.
- KUIP85 B.Kuipers, "Qualitative Simulation of Mechanisms," Technical Memo MIT/LCS/TM-274, Massachusetts Institute of Technology, USA, April 1985.
- KUIP84 B.Kuipers, "Commonsense Reasoning about Causality : Deriving Behaviour from Structure," *Artificial Intelligence*, v24, 169–204, December 1984.
- LEE.87 Y.C.Lee, K.S.Fu, "Machine Understanding of CSG : Extraction and Unification of Manufacturing Features," *IEEE Computer Graphics and Applications*, v2, n1, 20–32, January 1987.

- LEE_84 D.T.Lee, F.Preparata, "Computational Geometry : a Survey," *IEEE Trans. Computers*, vC-33, n12, 1072-1101, December 1984.
- LEE_83 D.T.Lee, "On Finding the Convex Hull of a Simple Polygon," *Int. J. Computer and Information Sciences*, v12, n2, 87-98, April 1983.
- LEWI81 W.P.Lewi, "The Role of Intelligence in the Design of Mechanical Components," in *Man-Machine Communication in CAD/CAM, IFIP WC5.2*, T.SATA, E.Warman(Ed), North Holland, 1981.
- LOZA83 T.Lozano-Perez, "Spatial Planning: a Configuration Space Approach," *IEEE Trans. Computers*, vC-32, n2, 108-120, February 1983.
- LOZA81 T.Lozano-Perez, "Automatic Planning of Manipulator Transfer Movements," *IEEE Trans.Sys.Man Cyb.*, v11, n10, 681-689, October 1982.
- MACC85 K.J.MacCallum, A.Duffy, "An Expert System for Preliminary Numerical Design Modelling," in *Engineering Software IV Proceedings of the 4th annual Conference, London*, 4-33-4-47, June 1985.
- McDE78 D.McDermott, "Circuit Design as Problem Solving," in *IFIP WC5.2 : Artificial Intelligence and Pattern Recognition in Computer Aided Design*, J.C.Latombe(Ed), North Holland, 227-245, 1978.
- MEDL86 A.J.Medland, "The Computer Based Design Process," Kogan Page, 1986.
- MILL85a A.S.Miller, R.G.Hannam, "Computer Aided Design Using a Knowledge Based Approach and its Application to the Design of Jigs and Fixtures," *Proc. Instn. Mech. Engrs. Part B*, v199, nB4, 227-234, 1985.
- MITT86 S.Mittal, C.L.Dym, M.Morjaria, "PRIDE : an Expert System for the Design of Paper Handling Systems," *IEEE Computer*, v19, n7, 102-114, July 1986.
- MOGE86 T.Mogensen, "The Application of Partial Evaluation to Ray-Tracing," Report of the Institute of Datalogy, University of Copenhagen, 1986.
- MOST85 J.Mostow, "Toward Better Models of the Design Process," *The AI Magazine*, v6, n1, 44-56, Spring 1985.
- OHSU83 Setsuo Ohsuga, "A New Method of Model Description — Use of Knowledge Bases and Inference," in *IFIP WC5.2 : CAD Systems Framework*, K.Bo, F.M.Lillehagen(Ed), North Holand, 1983.

- PAHL85 G.Pahl, "Denkpsychologische Erkenntnisse und Folgerungen Fuer die Konstruktionslehre," in *Proceedings of the International Conference on Engineering Design ICED 85 : Theory and Practice of Engineering Design in International Comparison*, V.Hubka(Ed), Edition Heuristo (Switzerland), 817–832 (Vol.2), August 1985.
- PAHL84 G.Pahl, W.Beitz, "Engineering Design," K.M.Wallace(Ed), Design Council (Springer Verlag), 1984.
- POPP86 R.J.Popplestone, T.Smithers, J.Corney, A.Koutsou, K.Millington, G.Sahar, "Engineering Design Support Systems," Dept. of A.I. Research Paper No 286, Edinburgh University, GB, 1986.
- PREN79 W.Prenowitz, J.Jantosciak, "Join Geometries : a Theory of Convex Sets and Linear Geometry," Springer Verlag, 1979.
- PREP77 F.P.Preparata, S.J.Hong, "Convex Hulls of Finite Sets of Points in 2 and 3 Dimensions," *Communications of the ACM*, v20, n2, 87–93, February 1977.
- REQU83 A.A.G.Requicha, H.B.Voelcker, "Solid Modelling : Current Status and Research Directions," *IEEE Computer Graphics and Applications*, 25–37, October 1983.
- REQU80a A.A.G.Requicha, "Representations of Rigid Solid Objects," in *Lecture Notes in Computer Science 89 : Computer Aided Design, Modelling, Systems Engineering, C.A.D. Systems.*, J.Encarnacao(Ed), Springer Verlag, 2–78, September 1980.
- REQU80b A.A.G.Requicha, "Representations for Rigid Solids : Theory, Methods and Systems.," *ACM Computing Surveys*, v12, n4, 437–464, December 1980.
- REYN84 M.Reynier, J.Fouet, "Automated Design of Crankcases : the CARTER system," *Computer Aided Design*, v16, n6, 308–313, November 1984.
- ROCK72 R.T.Rockafellar, "Convex Analysis," Princeton Univ. Press, 1972.
- ROSS86 J.R.Rossignac, A.A.G.Requicha, "Offsetting Operations in Solid Modelling," *Computer Aided Geometric Design*, North Holland, v3, n2, 129–148, August 1986.
- ROTH85 K.Roth, "Strategien zur Methodischen Entwicklung von Funktionsvarianten, prinzipiellen Effect- und Gestaltloesungen," in *Proceedings of the International Conference on Engineering Design ICED 85 : Theory and Practice of Engineering Design in International Comparison*, V.Hubka(Ed), Edition Heuristo (Switzerland), 278–294 (Vol.1), August 1985.

- RUNC84 C.Runciman, K.Swift, "Knowledge Based C.A.D. and Design for Economic Manufacture," in *Proceedings of the International Conference on Artificial Intelligence, Marseille, 15-23, October 1984*.
- SAMB83 A.S.Sambura, J.S.Gero, "Framework for a Computer Integrated Design Environment," in *IFIP WC5.2 : CAD Systems Framework*, K.Bo, F.M.Lillehagen(Ed), North Holland, 1983.
- SIX_82 H.Six, D.Wood, "Counting and Reporting Intersections of d-Ranges," *IEEE Transactions on Computers*, vC-31, n3, 181-187, March 1982.
- STOE70 W.Stoer, C.Witzgall, "Convexity and Optimisation in Finite Dimensions I," *Grundlehren der Mathematischen Wissenschaften Vol.163*, Springer Verlag, 1970.
- SWIF85 K.G.Swift, A.Matthews, P.Firth, C.S.Syan, "Artificial Intelligence in Engineering Design," *Man or Machine — A Choice of Intelligence, International Conference, Florida, CAM-I Inc. P-84-MM-04*, November 1984.
- SWIF83 K.Swift, A.Matthews, "Expert Computer Systems in Engineering Design," *Engineering*, v223, n9, 673-678, September 1983.
- TARS51 A.Tarski, "A Decision Method for Elementary Algebra and Geometry," University of California Press, 2nd Edition 1951.
- THOM80 R.Thomson, "Principles of Graphic Communication for Engineers," Thomas Nelson and Sons Ltd, 1980.
- ULLM87 D.G.Ullman, T.A.Dietterich, "Mechanical Design Methodology : Implications on Future Developments of Computer-Aided Design and Knowledge Based Systems," *Engineering with Computers*, v2, n1, 21-29, 1987.
- VALE64 F.Valentine, "Convex Sets," McGraw Hill, 1964.
- VOGW86 J.Vogwell, S.J.Culley, "Optimal Component Selection Using Engineering Databases," School of Engineering, University of Bath, UK, (private communication), 1986.
- WARM78 E.A.Warman, "Computer Aided Design : an Intersection of Ideas," in *IFIP TC-5 Working Conference on Artificial Intelligence and Pattern Recognition in Computer Aided Design, Grenoble, J.C.Latombe(Ed)*, North Holland, 1-18, 1978.

APPENDIX A

CONVEX SET THEORY AND TOPOLOGY

This Appendix introduces some mathematical definitions from Convex Set Theory in Section 1 and from Topology in Section 2.

A.1 CONVEX SET THEORY

This Section introduces some of the definitions and properties of Convex Sets. The starting point is the introduction of a convex Join operator in Subsection 1.1 which allows the definition of a Convex Set in Subsection 1.2. Subsection 1.3 then considers the Interior, Closure and Boundary of a Convex Set, before Subsection 1.4 finally introduces the Krein-Milman Theorem.

A.1.1 Convex Join Operator

Definition of Join Operator. *The open convex Join of two distinct points p_1 and p_2 (denoted by $p_1 \cdot p_2$) is defined as the set of points, p , such that $p = \lambda p_1 + (1 - \lambda)p_2$ with $0 < \lambda < 1$*

$$p_1 \cdot p_2 \equiv \{q \mid \exists \lambda \in \mathbb{R} \ q = \lambda p_1 + (1 - \lambda)p_2 \text{ where } 0 < \lambda < 1\}$$

Note that $p_1 \cdot p_2$ does not include either of the points p_1 or p_2 ; in Euclidean space, the Join of p_1 and p_2 is simply the open line segment between p_1 and p_2 . Also the convex Join of a point p with itself is equal to itself by definition : $p \cdot p \equiv p$.

The idea of convex Join can be extended to several points : The ‘open convex Join’ of n distinct points $p_1, p_2, \dots, p_n$, is simply defined as the set of points, p , such that $p = \lambda_1 p_1 + \lambda_2 p_2 + \dots + \lambda_n p_n$ where $\lambda_1 + \lambda_2 + \dots + \lambda_n = 1$.

Definition of Convex Hull. *The open convex Join of a set S , $S \cdot S$, is defined as the union of all open convex Joins of all subsets of S . For any set S , the set $S \cdot S$ is called the convex hull of S .*

The concept of convex Join can also be formulated in a more abstract manner, without appealing to notions of arithmetic or dimension, in terms of an operator (on points or sets of points) which fulfils certain axioms. Such a treatment is given in [PREN79] and forms the basis of *Join Geometries*.

Definition of Join Geometry. *A Join Geometry consists of a pair $(J, \cdot)$ where J is a set and $\cdot$ an operation which maps points of J into points of J and satisfies axioms J1–J7.*

Let a, b, c and d be points of J .

J1 (Existence law). $a \cdot b \neq \emptyset$.

J2 (Commutative law). $a \cdot b = b \cdot a$.

J3 (Associative law). $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.

J4 (Idempotent law). $a \cdot a = a$.

J5 (Existence law). $a/b \neq \emptyset$.

J6 (Four Term Transposition law). If $a/b \cap c/d \neq \emptyset$ then $a \cdot d \cap b \cdot c \neq \emptyset$.

J7 (Idempotent law). $a/a = a$.

Note that $/$ is the extension operator derived from $\cdot$ and is defined as follows.

Definition of the Extension Operation. *The extension of a point a from a point b of J , denoted by a/b , is the set of all points x of J which satisfy $a \subset b \cdot x$.*

A.1.2 Definition Of Convex Set

Definition of Convex Set. *A set, S , of points is convex if and only if (iff) for all points p, q of S , $p \cdot q$ is included in S .*

$$S \text{ Convex} \iff \forall p, q \in S \ p \cdot q \subset S$$

Some properties of convex sets are :

- The empty set, $\emptyset$, is convex because it satisfies the condition vacuously as it does not have any elements.
- Any set consisting of a single point, p , is also convex.
- The intersection of convex sets is also a convex set.

For proofs and a more comprehensive list of properties see [PREN79, VALE64 and EGGL69].

Definition of Polytope. *A polytope is a convex set which is the convex hull of a finite, nonempty set.*

A.1.3 Interior, Closure and Boundary of a Set

Definition of Interior Point. *A point, p , is an interior point of a convex set S iff for all points x of S , there exists at least one point y of S such that p is an element of $x \cdot y$.*

The set of all such points p is called the ‘interior’ of set S , denoted by $I(S)$.

Some properties of interiors are :

- The interior of the empty set, $\emptyset$, is the empty set itself.
- The interior of a singleton $\{p\}$ is the set $\{p\}$ itself.
- The interior of a convex set is also a convex set.
- The monotonic property of interiors : Let a convex set S contain a convex set T and let set T meet the interior of set S . Then the interior of T is included in the interior of S .

For proofs and a more comprehensive list of properties see [PREN79, VALE64 and EGGL69].

Definition of Contact Point. *A point p is a contact point of a convex set S iff there exists a point x of S such that $p \cdot x$ is included in S .*

The set of all such points p is called the ‘closure’ of set S , denoted by $Cl(S)$. A set S is closed iff $S = Cl(S)$.

Some properties of closed sets are :

- The empty set is closed, $\emptyset = Cl(\emptyset)$.
- The singleton set $\{p\}$ is also closed, $\{p\} = Cl(\{p\})$.
- The closure of a convex set is also a convex set.
- The intersection of closed sets is a closed set.

For proofs and a more comprehensive list of properties see [PREN79, VALE64 and EGGL69].

Definition of Boundary Point. A point p is a boundary point of a convex set S iff p belongs to the closure of S , but p does not belong to the interior of S : $p \in B(S)$ iff $p \in Cl(S)$, $p \notin I(S)$.

The set of such points p is called the boundary of S and is denoted by $B(S)$.

Some properties of boundaries are :

- The boundary of the empty set is empty.
- The boundary of a singleton set $\{p\}$ is the empty set.
- The interior and boundary of a closed convex set S are a partition of set S : the intersection of S 's interior and boundary is empty, and the union of the interior and boundary is equal to S . Thus S , closed convex, means that $S = I(S) \cup B(S)$.

For proofs and a more comprehensive list of properties see [PREN79, VALE64 and EGGL69].

Note that the boundary of a convex set is taken *relative* to the dimension of the convex set *itself* and does not depend on the dimension of the space containing the convex set. For example the boundary of a line segment consists of its two end points, even when the line segment is in a space of dimension greater than one. Similarly the boundary of a flat disc is its circumscribing circle, even when the disc is in a space of dimension greater than two.

In effect a convex set is said to be closed when it contains all of its boundary points and is said to be open if it does not contain any of its boundary points.

Definition of Boundedness. A set S is said to be bounded if the least upper bound (supremum) of the distance between any two of its points (ie its diameter) is finite.

Some properties of bounded sets are :

- The intersection of bounded sets is bounded.

- If set S is closed and bounded, then $S \cdot S$ is closed and bounded (Theorem 10 in [EGGL69]).

Definition of Extreme Point. *A point, p , of a convex set S is said to be an ‘extreme point’ of S iff p is never contained in the open convex Join of two or more distinct points of S .*

$$p \text{ extreme} \iff \text{if } p \in x \cdot y \text{ then } x = y$$

Some properties of extreme points are :

- The boundary of a closed convex set S contains the extreme points of S provided S is not a point. [PREN79]
- In Euclidean space, every non-empty, bounded, closed convex set has at least one extreme point. (see Theorem 11.4 in [VALE64])

A.1.4 The Krein-Milman Theorem

The Krein-Milman Theorem. *In a Euclidean space a closed convex set is the closure of the convex hull of its extreme points (Theorem 13 in [EGGL69]). More generally, a convex, compact, non-empty set is equal to the convex closure of its extreme points (p174, [BERG66]).*

Proofs can be found on page 24 in [EGGL69], page 167 in [BERG63], Section 18 in [ROCK70], Sections 13.10 to 13.15 in [PREN79] and part XI in [VALE64]. (Compactness is defined in Section 2 of Appendix A).

Corollary. *Let X and Y be two closed, bounded convex sets in Euclidean space. Let X be non-empty, non-singleton. If $B(X)$ is included in a convex set Y , then X is included in Y .*

The proof follows from the Krein-Milman theorem : If $B(X)$ is included in Y , then all the extreme points of X are included in Y . As Y is convex, the convex hull of all extreme points of X is also included in Y , that is X is included in Y .

A.2 TOPOLOGY

This Section introduces a few definitions and properties from Topology. In particular the definitions of Topological Space and Compactness are followed by the compactification of the Real Number line and the Tychonoff Theorem.

Definition of Topological Space. *A nonempty set S together with a subcollection $\mathcal{T} \subset 2^S$ is a topological space if the following three axioms are satisfied :*

$$A1 \quad S \in \mathcal{T}, \emptyset \in \mathcal{T}.$$

$$A2 \quad T_1, T_2 \in \mathcal{T} \Rightarrow T_1 \cap T_2 \in \mathcal{T}.$$

$$A3 \quad \forall \mu \in M, T_\mu \in \mathcal{T} \Rightarrow \bigcup \{T_\mu \mid \mu \in M\} \in \mathcal{T}.$$

Note that 2^S denotes the power set of S , that is the set of all subsets of S , and M is an arbitrary set of integer subscripts.

Thus $\mathcal{T}$ is a set of subsets of S which satisfies the three conditions : both the empty set and S itself are included in $\mathcal{T}$, the intersection of any two members of $\mathcal{T}$ is a member of $\mathcal{T}$ and the union of the members of any subset of $\mathcal{T}$ is also a member of $\mathcal{T}$. The set $\mathcal{T}$ is called a topology of S . The elements of $\mathcal{T}$ are called 'open sets'. For further information see [KELL55] Chapter 1, [GAME83] Chapter 2.

Definition of Compactness. *A set S is compact if every open cover of S has a finite subcover. That is, S is compact if, whenever $\{U_\mu\}_{\mu \in M}$ is a family of open sets whose union is S , then there are finitely many of the U_μ 's whose union is S .*

Note that a collection of sets $\{U_\mu\}_{\mu \in M}$ is a cover of some set S if S is included in the union of the sets in the collection, that is $S \subset \bigcup_{\mu \in M} U_\mu$

Theorem. *Any closed set S contained in a compact set R is itself compact [Section IV.6 in both BERG63 and BERG66].*

Example. *Compactification of real Number line.*

The space (line) of all real numbers, $\mathbb{R}$, is not bounded as there is no largest (or smallest) real number. Although $\mathbb{R}$ can be covered with open intervals of unit length, no finite subcovering of $\mathbb{R}$ can be found. $\mathbb{R}$ is therefore not compact.

The real number space $\mathbb{R}$ can be extended by adding two elements called $+\infty$ and $-\infty$ and, for any $\lambda \in \mathbb{R}$, stipulating that

$$+\infty = (-\infty)(-\infty) = (+\infty)(+\infty) = \lambda + (+\infty) = \lambda - (-\infty)$$

$$-\infty = (-\infty)(+\infty) = (+\infty)(-\infty) = \lambda + (-\infty) = \lambda - (+\infty)$$

$$+(-\infty) = -(+\infty) = -\infty$$

$$-\infty < \lambda < +\infty$$

And further for any $\lambda > 0$:

$$(+\infty)\lambda = \lambda(+\infty) = (-\lambda)(-\infty) = (-\infty)(-\lambda) = +\infty$$

$$\frac{\lambda}{-\infty} = \frac{\lambda}{+\infty} = 0$$

The extended real number space is denoted by $\hat{\mathbb{R}}$. For further information see Section I.1 in both [BERG63] and [BERG66].

Now $(\hat{\mathbb{R}}, \mathcal{T})$ is a topological space. The collection of open sets $\mathcal{T}$ contains the open intervals of $\mathbb{R}$, the union of $\{+\infty\}$ with an open interval of $\mathbb{R}$ of the form $]\lambda, +\infty[$ and the union of $\{-\infty\}$ with an open interval of $\mathbb{R}$ of the form $]-\infty, \lambda[$.

Theorem. $\hat{\mathbb{R}}$ is a compact space.

For further information and the proof of compactness of $\hat{\mathbb{R}}$ see Section IV.6 in both [BERG63] and [BERG66].

Tychonoff Theorem. *Any product of compact spaces is compact.*

Proofs may be found in [GAME83] p102 and [KELL55] p143.

In particular the n -dimensional extended real number space $\hat{\mathbb{R}}^n = \overbrace{\hat{\mathbb{R}} \times \dots \times \hat{\mathbb{R}}}^{n \text{ times}}$ is therefore compact.

It follows that any closed, non-empty convex set S in $\hat{\mathbb{R}}^n$ is the convex hull of its extreme points in $\hat{\mathbb{R}}^n$. If some or all of the extreme points are at infinity (that is, have infinite coordinates) then S is of infinite size (that is, the least upper bound on its diameter is $+\infty$).

APPENDIX B

COMPARISON OF CONVEX SETS

This Appendix provides a very brief summary of a mathematical approach to the question of what it means to compare convex sets. The full proofs of the material presented in this Appendix are available on request from the author and consists of an internal working report of 152 pages.

B.1 SUMMARY

Section 2 of this Appendix both introduces the problem of defining a ‘comparison’ operation on sets and solves it in a manner that suggests the existence of exactly five distinct comparison outcomes. Section 3 then considers the relationship between term by term comparisons and the comparison of Cartesian product sets themselves which is shown to be representable as a simple semi-lattice. Section 4 considers the more specialised case of the comparison of convex sets where the original five comparison outcomes may be refined into twenty eight more detailed distinctions. Section 5 similarly refines the relationship between term by term comparisons and the comparison of Cartesian product sets themselves to convex sets. This relationship shown to be very nearly representable as a semi-lattice.

The results shown in this Appendix are significant as they show the increase in the complexity of structure which has to be expected in a system which allows the representation of whole sets rather than being limited to just single points as its primitive unit. It is interesting to note that in the case of points all the results

presented in this Appendix degenerate properly into the only two possibilities that exist for points : namely being either equal or distinct.

The increase in comparison outcomes observed as the result of considering convex sets flows directly from their increased internal structure : in particular, the existence of an 'interior' and a 'boundary'.

The significance of the result of Section 5 is that the comparison on a term by term basis is sufficient to determine the nature of the relationship between two Cartesian product sets. The practical application of this result is immediate in the case of comparing of two n -dimensional excursion boxes each of which is the Cartesian product of n intervals, that is, of n one-dimensional convex sets. Only the corresponding intervals need be compared and the results combined using the method described in Section 5. This greatly simplifies the computational task of comparing excursion boxes in n -dimensional space.

B.2 THE COMPARISON OF TWO SETS

What is meant by ‘the comparison of two sets’ ? Usually, the comparison of two ‘things’ involves an enumeration of those aspects in which the two ‘things’ are equal and of those in which they differ. Adapting this idea to sets leads to the following : The comparison two sets is equivalent to the enumeration of those elements which are common to both sets and of those which only occur in one of the sets. Expressed differently, it consists of looking at what the intersection of the two sets is equal to and what is not equal to. The possible outcomes of such an operation can be grouped into five distinct cases.

Theorem of Set Comparison. *Let U be the universal set and A and B be any two non-empty subsets of U . Compare (A, B) , the comparison of set A to set B , partitions $U \times U$ into the following five equivalence classes :*

$$\left\{ \begin{array}{ll} A \cap B = \emptyset, A \cap B \neq A, A \cap B \neq B & A \text{ and } B \text{ disjoint} & (1) \\ A \cap B \neq \emptyset, A \cap B \neq A, A \cap B \neq B & A \text{ intersects } B \text{ properly} & (2) \\ A \cap B \neq \emptyset, A \cap B \neq A, A \cap B = B & A \text{ properly contains } B & (3) \\ A \cap B \neq \emptyset, A \cap B = A, A \cap B \neq B & A \text{ is properly included in } B & (4) \\ A \cap B \neq \emptyset, A \cap B = A, A \cap B = B & A \text{ equals } B & (5) \end{array} \right.$$

where $A \cap B$ denotes the intersection between set A and set B , $\times$ the Cartesian product operator, and $\emptyset$ the empty set.

The explanation is as follows. In case 1 the two sets do not have any shared elements. Their intersection is therefore empty although neither set is empty. In case 2 the two sets do intersect and there are some elements of A which are also elements of B , but neither set is included in the other : there is at least one element of A which is not an element of B and vice versa. In case 3 set A contains set B , but set A also contains at least one element which is not in B : the set A is in some sense ‘bigger’ than B . In case 4 the set A is included in set B : all elements of set A are also elements of set B , but there is also at least one element in B which is not in A . This time B is in some sense ‘bigger’ than A .

Finally, case 5 corresponds to the two sets being equal : all elements of A are also elements of B and vice versa. These five cases are both mutually exclusive and exhaust all possibilities. These two facts are useful in some of the proofs.

Some of the comparison cases are symmetric in the sense that Compare (A to B) equals Compare (B to A), while others are not. In particular, the “disjoint”(1), “properly intersecting”(2) and “equals”(5) comparison outcomes are symmetric. The remaining comparison outcomes form an opposite pair in the sense that if Compare (A to B) is “contains”(3), then Compare (B to A) is “included in”(4) and vice versa.

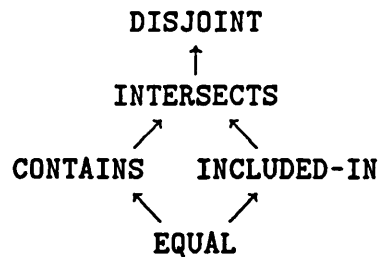
Note that the comparison of a non-empty set with the empty set has been excluded as being meaningless : As the empty set has no elements, comparing a non-empty set to it would reduce to enumerating all elements of the non-empty set as they are all different from nothingness.

B.3 THE COMPARISON OF CARTESIAN PRODUCT SETS

Definition of an Ordering on Set Comparison Outcomes. Let C denote the set of possible comparison outcomes, $C = \{\text{DISJOINT, INTERSECTS, CONTAINS, INCLUDED-IN, EQUAL}\}$ and $\geq$ the following (partial) ordering relationship on the elements of C :

$$\begin{aligned}
 \text{DISJOINT} &\geq k, \forall k \in C \\
 \text{INTERSECTS} &\geq k, \forall k \in C, k \neq \text{DISJOINT} \\
 \text{CONTAINS} &\geq k, \forall k \in \{\text{CONTAINS, EQUAL}\} \\
 \text{INCLUDED-IN} &\geq k, \forall k \in \{\text{INCLUDED-IN, EQUAL}\} \\
 \text{EQUAL} &\geq \text{EQUAL}
 \end{aligned}$$

This may be represented pictorially as



most dominant

least dominant

The partial ordering on C is such that every pair of elements of C has a unique least upper bound (supremum); in other words the elements of C form an upper semi-lattice.

Note that each element of C is its own greatest upper bound. Also note that with partial ordering defined above, each pair of elements of C has a greatest lower bound (infimum). This in fact makes C into a lattice, that is a partially ordered set in which each pair of elements has both a supremum and an infimum.

Theorem Structure of Set Comparison Outcomes. *Let $\vee$ denote the least upper bound (supremum) operator on elements of C , then $(C, \vee)$ is an Abelian monoid.*

That $(C, \vee)$ is an Abelian monoid simply means that the supremum operation has the following properties : let $A, B, D, \dots$, etc all be arbitrary elements of C , then

- the supremum operation is commutative (Abelian) as $A \vee B = B \vee A$.
- the supremum operation is associative as $A \vee (B \vee D) = (A \vee B) \vee D$.
This means that the order in which the least upper bounds are evaluated in expressions of the form $A_1 \vee A_2 \vee \dots \vee A_n$ is not important; whatever order is chosen, the answer will be the same.
- the supremum operation has an identity element (EQUAL) as the outcome of $A \vee \text{EQUAL}$ and $\text{EQUAL} \vee A$ is always A .

Theorem of Cartesian Product Set Comparison. *Let $\times$ denote the Cartesian product operator, and let A_f be a family of n sets $A_1, A_2, \dots, A_n$ and similarly B_f a family of n sets $B_1, B_2, \dots, B_n$. Let $A_\times$ and $B_\times$ be the Cartesian products of the families A_f and B_f :*

$$A_\times = A_1 \times A_2 \times \dots \times A_n$$

$$B_\times = B_1 \times B_2 \times \dots \times B_n$$

The outcome of the comparison of $A_\times$ and $B_\times$ is the least upper bound (supremum) of the n term by term comparisons between the A_i and B_i for $i = 1, \dots, n$:

$$\begin{aligned}
\text{Compare}(A_{\times}, B_{\times}) &= \text{Compare}(A_1, B_1) \vee \dots \vee \text{Compare}(A_n, B_n) \\
&= \bigvee_{i=1, n} \{\text{Compare}(A_i, B_i)\}
\end{aligned}$$

Thus outcome of the comparison of two n -term Cartesian product sets is the least upper bound (supremum) of the ordering relation on the n outcomes of the product component by component comparisons.

For example consider a set $A_{\times}$ which is the Cartesian product of two sets A_1 and A_2 , and a set $B_{\times}$ which is the Cartesian product of B_1 and B_2 : $A_{\times} = A_1 \times A_2$ and $B_{\times} = B_1 \times B_2$. If A_1 properly contains B_1 and A_2 is properly included in B_2 then how does $A_{\times}$ compare to $B_{\times}$?. Using the above results :

$$\begin{aligned}
\text{Compare}(A_{\times}, B_{\times}) &= \text{Compare}(A_1, B_1) \vee \text{Compare}(A_2, B_2) \\
&= \text{CONTAINS} \vee \text{INCLUDED-IN} \\
&= \text{INTERSECTS}
\end{aligned}$$

Thus $A_{\times}$ and $B_{\times}$ are a pair of properly intersecting sets.

B.4 THE COMPARISON OF CONVEX SETS

For a definition of convex sets and some of their properties see Appendix A.

An important property of closed convex sets is that they possess an interior and a boundary. In particular the interior and boundary of a convex set S are disjoint and their union is equal to S , that is, $S = I(S) \cup B(S)$ where $I(S)$ denotes the interior of S , $B(S)$ the boundary of S and $\cup$ the set theoretic union operator.

Let $A = I(A) \cup B(A)$ and $B = I(B) \cup B(B)$ be two closed, bounded, non-empty, convex sets. Their intersection $A \cap B$ can be expressed as

$$\begin{aligned} A \cap B &= \{I(A) \cup B(A)\} \cap \{I(B) \cup B(B)\} \\ &= \{I(A) \cap I(B)\} \cup \{I(A) \cap B(B)\} \\ &\quad \cup \{B(A) \cap I(B)\} \cup \{B(A) \cap B(B)\} \end{aligned}$$

Thus to identify the possible outcomes of a comparison between two closed bounded convex sets A and B we have to consider all four of the above intersection terms (which incidentally are mutually disjoint).

Theorem of Convex Set Comparison. *Let U be the universe of all closed, bounded, convex sets in a finite dimensional Euclidean space. Comparing a non-empty, closed, bounded, convex set A to a non-empty, closed, bounded, convex set B partitions $U \times U$ into 28 equivalence classes which can be grouped under the 5 headings : A, B Disjoint (1 case), A Intersects B (13 cases), A Contains B (6 cases), A Included in B (6 cases), A Equals B (2 cases). These cases are tabulated below :*

Disjoint	$I(A) \cap I(B)$	$I(A) \cap B(B)$	$B(A) \cap I(B)$	$B(A) \cap B(B)$
DISJOINT	$\neq I(A)$	$\neq I(A)$	$\neq B(A)$	$\neq B(A)$
	$\neq I(B)$	$\neq B(B)$	$\neq I(B)$	$\neq B(B)$
	$= \emptyset$	$= \emptyset$	$= \emptyset$	$= \emptyset$

Intersect	$I(A) \cap I(B)$	$I(A) \cap B(B)$	$B(A) \cap I(B)$	$B(A) \cap B(B)$
all	$\neq I(A)$	$\neq I(A)$	$\neq B(A)$	$\neq B(A)$
all	$\neq I(B)$	$\neq B(B)$	$\neq I(B)$	$\neq B(B)$
INTER1	$= \emptyset$	$= \emptyset$	$= \emptyset$	$\neq \emptyset$
INTER2	$= \emptyset$	$= \emptyset$	$\neq \emptyset$	$= \emptyset$
INTER3	$= \emptyset$	$= \emptyset$	$\neq \emptyset$	$\neq \emptyset$
INTER4	$= \emptyset$	$\neq \emptyset$	$= \emptyset$	$= \emptyset$
INTER5	$= \emptyset$	$\neq \emptyset$	$= \emptyset$	$\neq \emptyset$
INTER6	$\neq \emptyset$	$= \emptyset$	$= \emptyset$	$= \emptyset$
INTER7	$\neq \emptyset$	$= \emptyset$	$= \emptyset$	$\neq \emptyset$
INTER8	$\neq \emptyset$	$= \emptyset$	$\neq \emptyset$	$= \emptyset$
INTER9	$\neq \emptyset$	$= \emptyset$	$\neq \emptyset$	$\neq \emptyset$
INTER10	$\neq \emptyset$	$\neq \emptyset$	$= \emptyset$	$= \emptyset$
INTER11	$\neq \emptyset$	$\neq \emptyset$	$= \emptyset$	$\neq \emptyset$
INTER12	$\neq \emptyset$	$\neq \emptyset$	$\neq \emptyset$	$= \emptyset$
INTER13	$\neq \emptyset$	$\neq \emptyset$	$\neq \emptyset$	$\neq \emptyset$

Contains	$I(A) \cap I(B)$	$I(A) \cap B(B)$	$B(A) \cap I(B)$	$B(A) \cap B(B)$
all	$\neq I(A)$	$\neq I(A)$	$\neq B(A)$	$\neq B(A)$
CONT1	$= I(B)$	$= B(B)$	$\neq I(B)$	$= B(B)$
	$\neq \emptyset$	$= \emptyset$	$= \emptyset$	$= \emptyset$
CONT2	$\neq I(B)$	$= B(B)$	$= I(B)$	$= B(B)$
	$= \emptyset$	$= \emptyset$	$\neq \emptyset$	$= \emptyset$
CONT3	$= I(B)$	$= B(B)$	$\neq I(B)$	$\neq B(B)$
	$\neq \emptyset$	$\neq \emptyset$	$= \emptyset$	$= \emptyset$
CONT4	$= I(B)$	$\neq B(B)$	$\neq I(B)$	$= B(B)$
	$\neq \emptyset$	$= \emptyset$	$= \emptyset$	$\neq \emptyset$
CONT5	$= I(B)$	$\neq B(B)$	$\neq I(B)$	$\neq B(B)$
	$\neq \emptyset$	$\neq \emptyset$	$= \emptyset$	$\neq \emptyset$
CONT6	$\neq I(B)$	$\neq B(B)$	$= I(B)$	$= B(B)$
	$= \emptyset$	$= \emptyset$	$\neq \emptyset$	$\neq \emptyset$

Included	$I(A) \cap I(B)$	$I(A) \cap B(B)$	$B(A) \cap I(B)$	$B(A) \cap B(B)$
all	$\neq I(B)$	$\neq B(B)$	$\neq I(B)$	$\neq B(B)$
INCL1	$= I(A)$	$\neq I(A)$	$= B(A)$	$= B(A)$
	$\neq \emptyset$	$= \emptyset$	$= \emptyset$	$= \emptyset$
INCL2	$\neq I(A)$	$= I(A)$	$= B(A)$	$= B(A)$
	$= \emptyset$	$\neq \emptyset$	$= \emptyset$	$= \emptyset$
INCL3	$= I(A)$	$\neq I(A)$	$= B(A)$	$\neq B(A)$
	$\neq \emptyset$	$= \emptyset$	$\neq \emptyset$	$= \emptyset$
INCL4	$= I(A)$	$\neq I(A)$	$\neq B(A)$	$= B(A)$
	$\neq \emptyset$	$= \emptyset$	$= \emptyset$	$\neq \emptyset$
INCL5	$= I(A)$	$\neq I(A)$	$\neq B(A)$	$\neq B(A)$
	$\neq \emptyset$	$= \emptyset$	$\neq \emptyset$	$\neq \emptyset$
INCL6	$\neq I(A)$	$= I(A)$	$\neq B(A)$	$= B(A)$
	$= \emptyset$	$\neq \emptyset$	$= \emptyset$	$\neq \emptyset$

Equals	$I(A) \cap I(B)$	$I(A) \cap B(B)$	$B(A) \cap I(B)$	$B(A) \cap B(B)$	
EQUAL1	$= I(A)$	$\neq I(A)$	$\neq B(A)$	$= B(A)$	(A,B points)
	$= I(B)$	$\neq B(B)$	$\neq I(B)$	$= B(B)$	
	$\neq \emptyset$	$= \emptyset$	$= \emptyset$	$= \emptyset$	
EQUAL2	$= I(A)$	$\neq I(A)$	$\neq B(A)$	$= B(A)$	
	$= I(B)$	$\neq B(B)$	$\neq I(B)$	$= B(B)$	
	$\neq \emptyset$	$= \emptyset$	$= \emptyset$	$\neq \emptyset$	

These 28 cases will, hereafter, be referred to by their names DISJOINT, INTER1, INTER2, ..., INTER13, CONT1, ..., CONT6, ... etc.

The large number of cases results partly from having to deal with degenerate convex sets which contain only one element and whose interior is therefore a single point while the boundary is empty.

Note that these 28 possible comparison outcomes are mutually disjoint, exhaust all possibilities and are independent of the idea of dimension. They therefore apply to convex sets of arbitrary finite dimension.

B.5 THE COMPARISON OF CARTESIAN PRODUCTS OF CONVEX SETS

Definition of a Pseudo-Ordering on Convex Set Comparison Outcomes. Let K denote the set of possible comparison outcomes between convex sets, $K = \{\text{DISJOINT}, \text{INTER1}, \text{INTER2}, \text{INTER3}, \text{INTER4}, \text{INTER5}, \text{INTER6}, \text{INTER7}, \text{INTER8}, \text{INTER9}, \text{INTER10}, \text{INTER11}, \text{INTER12}, \text{INTER13}, \text{CONT1}, \text{CONT2}, \text{CONT3}, \text{CONT4}, \text{CONT5}, \text{CONT6}, \text{INCL1}, \text{INCL2}, \text{INCL3}, \text{INCL4}, \text{INCL5}, \text{INCL6}, \text{EQUAL1}, \text{EQUAL2}\}$ and $\geq$ the following partial pseudo-ordering relationship on the elements of K :

$$\text{DISJOINT} \geq \text{INTER1}$$

$$\begin{aligned} \text{INTER1} &\geq \{\text{INTER3}, \text{INTER5}\} \\ \text{INTER2} &\geq \{\text{INTER8}, \text{CONT2}\} \\ \text{INTER3} &\geq \{\text{INTER2}, \text{INTER13}, \text{CONT6}\} \\ \text{INTER4} &\geq \{\text{INTER10}, \text{INCL2}\} \\ \text{INTER5} &\geq \{\text{INTER4}, \text{INTER13}, \text{INCL6}\} \\ \text{INTER6} &\geq \{\text{CONT1}, \text{INCL1}\} \\ \text{INTER7} &\geq \{\text{INTER6}, \text{CONT4}, \text{INCL4}\} \\ \text{INTER8} &\geq \{\text{INTER6}, \text{INCL3}\} \\ \text{INTER9} &\geq \{\text{INTER7}, \text{INTER8}, \text{INCL5}\} \\ \text{INTER10} &\geq \{\text{INTER6}, \text{CONT3}\} \\ \text{INTER11} &\geq \{\text{INTER7}, \text{INTER10}, \text{CONT5}\} \\ \text{INTER12} &\geq \text{INTER6} \\ \text{INTER13} &\geq \{\text{INTER9}, \text{INTER11}, \text{INTER12}\} \end{aligned}$$

$$\begin{aligned} \text{CONT1} &\geq \text{EQUAL1} & \text{INCL1} &\geq \text{EQUAL1} \\ \text{CONT2} &\geq \text{CONT1} & \text{INCL2} &\geq \text{INCL1} \\ \text{CONT3} &\geq \text{CONT1} & \text{INCL3} &\geq \text{INCL1} \\ \text{CONT4} &\geq \{\text{CONT1}, \text{EQUAL2}\} & \text{INCL4} &\geq \{\text{INCL1}, \text{EQUAL2}\} \\ \text{CONT5} &\geq \{\text{CONT3}, \text{CONT4}\} & \text{INCL5} &\geq \{\text{INCL3}, \text{INCL4}\} \\ \text{CONT6} &\geq \{\text{CONT2}, \text{CONT5}\} & \text{INCL6} &\geq \{\text{INCL2}, \text{INCL5}\} \end{aligned}$$

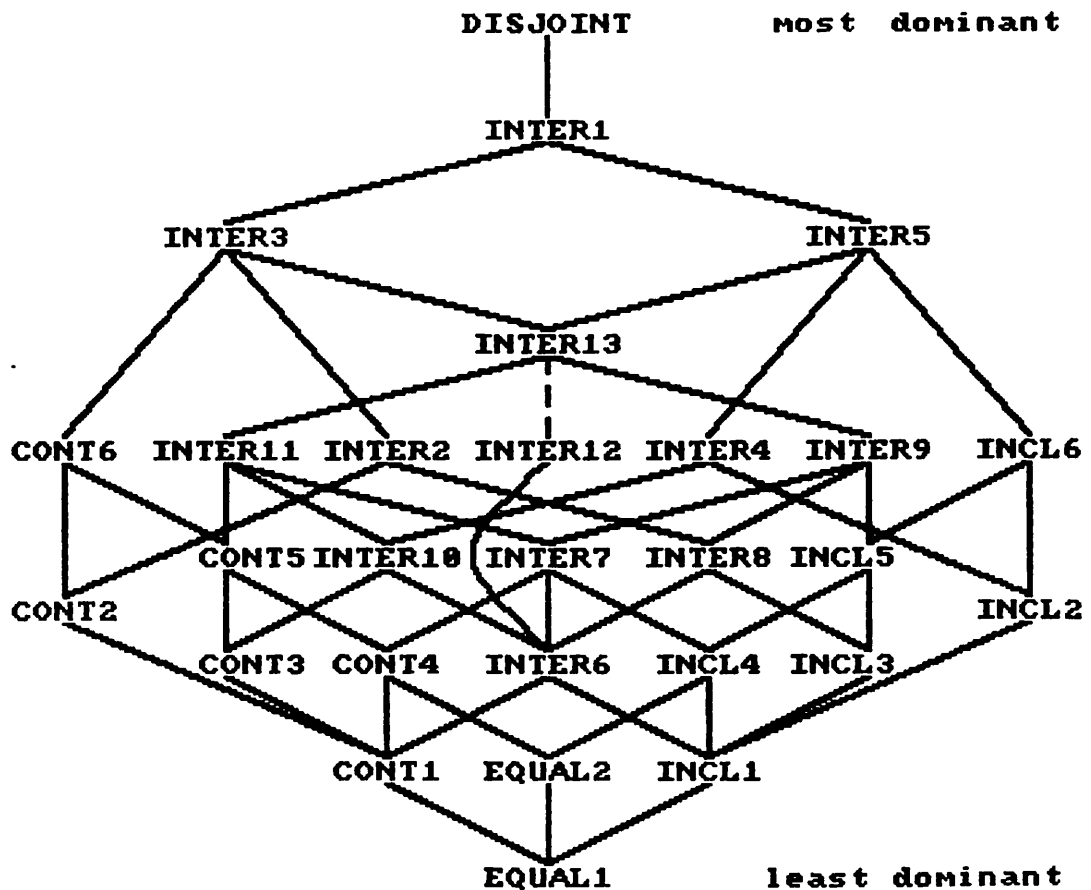
$$\begin{aligned} \text{EQUAL2} &\geq \text{EQUAL1} \\ \text{EQUAL1} &\geq \emptyset \end{aligned}$$

Transitivity applies, but has been omitted for clarity. Each comparison outcome also in relation with itself—with the exception of INTER12 : INTER12 is only in

relation with itself on the condition that a unique INTER12 outcome is present. If it were not for the exceptional behaviour of INTER12, the $\geq$ relation would be a partial order. Therefore the $\geq$ relation has been termed a partial pseudo-order.

Nevertheless the partial pseudo-ordering on K is such that every pair of elements of K has a unique least upper bound (supremum) in K .

The partial pseudo-ordering amongst the possible comparison outcomes can be represented pictorially :



$$B_{\times} = B_1 \times B_2 \times \dots \times B_n$$

The outcome of the comparison of $A_{\times}$ and $B_{\times}$ is the least upper bound (supremum) of the n term by term comparisons between the A_i and B_i for $i = 1, \dots, n$:

$$\begin{aligned} \text{Compare}(A_{\times}, B_{\times}) &= \text{Compare}(A_1, B_1) \vee \dots \vee \text{Compare}(A_n, B_n) \\ &= \bigvee_{i=1, n} \{\text{Compare}(A_i, B_i)\} \end{aligned}$$

Thus the comparison of two Cartesian product sets is equal to the least upper bound of the comparisons of the corresponding terms in the Cartesian product. The only restriction being associated with the intersection case 12 (INTER12) which is its own least upper bound only on condition of there being a unique INTER12 case amongst the component comparisons—otherwise INTER13 becomes the required least upper bound.

The exceptional behaviour of the INTER12 case can be explained as follows. The INTER12 comparison outcome between two convex sets only ever occurs in the case of two properly overlapping intervals; for example [1.6, 6.9] and [3.7, 100.2]. The important property is that the concerned convex sets (intervals) are one-dimensional. The boundary of a one-dimensional convex set (interval) consists of the two end points of the interval and is therefore disjoint. However, the boundary of a two- or higher-dimensional convex set is connected. The exceptional behaviour of the INTER12 comparison outcome essentially reflects this change in the properties of the boundary when the dimension of the concerned convex sets increases as the result of Cartesian products with other convex sets.

The consequences of the above theorem is that the comparison of excursion boxes in n -dimensional space can be achieved by first comparing their extents along the axis of the Cartesian coordinate system of the surrounding space and then combining these comparison outcomes according to the rule given in the above Theorem.

For example consider a set $A_{\times}$ which is the Cartesian product of two sets A_1 and A_2 , and a set $B_{\times}$ which is the Cartesian product of B_1 and B_2 : $A_{\times} = A_1 \times A_2$ and $B_{\times} = B_1 \times B_2$.

Example 1.

If $\text{Compare}(A_1, B_2) = \text{CONT1}$ and $\text{Compare}(A_1, B_2) = \text{INCL1}$ then how does $A_{\times}$ compare to $B_{\times}$?. Using the above results :

$$\begin{aligned}\text{Compare}(A_{\times}, B_{\times}) &= \text{Compare}(A_1, B_1) \vee \text{Compare}(A_2, B_2) \\ &= \text{CONT1} \vee \text{INCL1} \\ &= \text{INTER6}\end{aligned}$$

Example 2.

If $\text{Compare}(A_1, B_2) = \text{INTER12}$ and $\text{Compare}(A_1, B_2) = \text{INTER12}$ then how does $A_{\times}$ compare to $B_{\times}$?. Using the above results :

$$\begin{aligned}\text{Compare}(A_{\times}, B_{\times}) &= \text{Compare}(A_1, B_1) \vee \text{Compare}(A_2, B_2) \\ &= \text{INTER12} \vee \text{INTER12} \\ &= \text{INTER13}\end{aligned}$$