M

Requirements and Architecture of a CAM-oriented CAD System for Design and Manufacture of Mechanical Parts

by

Farhad Arbab

CSD 821201

Computer Science Department
University of Los Angeles
Los Angeles
1982

CSD 821201

UNIVERSITY OF CALIFORNIA
Los Angeles

Requirements and Architecture of a CAM-oriented CAD System for Design and Manufacture of Mechanical Parts

A dissertation submitted in partial fulfillment of the requirements for the degree Doctor of Philosophy in Computer Science

by

Farhad Arbab

1982

The dissertation of Farhad Arbab is approved.

[illegible]

Daniel M. Berry

[illegible]

David G. Cantor

[illegible]

Robert M. Hayes

David F. Martin

David F. Martin

Barrett O'Neill

Barrett O'Neill

[illegible]

Michel A. Melkanoff, Committee Chair

University of California, Los Angeles

1982

In her memory,
whose unending love and concern I miss the most,
to all those who taught me
and made learning worthwhile,
my parents,
my wife,
my friends,
and my teachers.

CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to express my appreciation for the kind assistance of many people who helped me during the course of this research. First, I am thankful to my advisor, Professor Michel A. Melkanoff, for his continued support and guidance, and for his sincere personal concern. I am indebted to my friend Larry Lichten, for our numerous discussions, his careful reading of my drafts, and his excellent remarks and suggestions. I am grateful to the members of my committee, Professors Daniel M. Berry (Computer Science), Robert M. Hayes (Dean, Library and Information Science), David F. Martin (Computer Science), and Barrett O'Neill (Mathematics), for their advice and cooperation, with my special thanks to Professor David G. Cantor (Mathematics) for his candid involvement and kind guidance.

I would also like to thank Vince Kelley for his comments on my final draft, Ron Eaton and Paolo Roberti for their input on UCLA's CAD Laboratory systems, Robert Chang, Esfandiar Kamvar, and the rest of my friends at the CAD/CAM Group for their direct and indirect assistance. I extend my regards to Richard Hull (University of Southern California) and Richard Wurtzel (Lockheed-California Company) for our fruitful discussions. It is my pleasure to express my gratitude to Sally Chan for her part in drawing of figures, to Bijan Arbab for his brotherly help and drawing of figures, and to Jamie Brody, Rosetta Lindsey, and my mother, Ozra Gordji, who helped with typing at various stages of this work. This research was made possible in part by funds from IBM

University Support Program and by a fellowship grant from the Society of Manufacturing Engineers Foundation, whose support is acknowledged.

Finally, I am grateful to my wife, Hamideh, whose love and companionship I cherish, for the many late-night hours she spent drawing, cutting, pasting, and checking, for her endless support and encouragement, and most of all for her endurance.

VITA

April 29, 1952 - Born, Zanjan, Iran

1973-1976 - IBM World Trade Corporation - Iranian Branch

1976 - B.S., Chemical Engineering
Tehran (Arya-Mehr) University of Technology

1976-1977 - Control Data Corporation - Middle East Branch

1977 - M.S., Computer Science
Tehran (Arya-Mehr) University of Technology

1978-1981 - IBM Los Angeles Scientific Center

1980-1982 - Teaching Fellow and Post Graduate Research Engineer
Computer Science Department,
University of California, Los Angeles

1982 - Ph.D., Computer Science
University of California, Los Angeles

PUBLICATIONS

Arbab, F.
"Notes on The Semantics and Optimization of a VHLL"
IBM Los Angeles Scientific Center
Report No. G320-2706, September 1980

Melkanoff, M. A., Lichten, L., and Arbab, F.
"An Overview of CAD/CAM
with Special Emphasis on Mechanical Applications"
The UCLA Computer Science Department Quarterly, Vol 9 No 4, Fall 1981

Lichten, L., Melkanoff, M. A., and Arbab, F.
"CAD/CAM"
Perspective, Vol 6 No 1, January 1982
Office of Academic Computing, UCLA

Arbab, F., Cantor, D. G., Lichten, L., and Melkanoff, M. A.
"The DSG CAM-Oriented Modeling System"
Proc. MIT Conf. on CAD/CAM Technology in Mechanical Engineering,
March 1982

Arbab, F., Cantor, D. G., Lichten, L., and Melkanoff, M. A.
"DSG: A Manufacturing-Oriented Approach to Solid Modeling"
Proc. CAM-I Meeting, Long Beach, April 1982

Arbab, F., Lichten, L., and Melkanoff, M. A.
"DSG: A CAM-Oriented Solid Modeling Interface"
Proc. Graphics Interface 82, Toronto, May 1982

Arbab, F., Lichten, L., and Melkanoff, M. A.
"An Approach to CAM-Oriented CAD"
Proc. ACM-IEEE Design Automation Conference, June 1982

ABSTRACT OF THE DISSERTATION

Requirements and Architecture
of a CAM-oriented CAD System for Design and Manufacture of
Mechanical Parts

by

Farhad Arbab
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 1982
Professor Michel A. Melkanoff, Chair

This work focuses on the Computer Aided Design and Computer Aided Manufacturing (CAD/CAM) of mechanical parts and on the advantage of an approach to solid definition based on a three-dimensional geometric model that considers manufacturing processes _during_ the design phase. We refer to this as CAM-oriented CAD methodology. Much effort is being spent on establishing the cliched "bridge between CAM and CAD". In pursuit of integrated CAD/CAM systems, we believe that our CAM-oriented approach to CAD promises significant benefits in the design-to-manufacture cycle of production.

Geometric solid modeling is a fundamental requirement for CAD/CAM systems that deal with mechanical parts. Several such modeling schemes have been proposed and are used in existing systems. This work includes an extended introduction to several commercially available and experimental systems, and reviews popular solid representation schemes, discussing their potentials as candidate models for integrated CAD/CAM systems. A new mathematical formalism for modeling of solid, rigid bodies, called Realizable Shape Calculus (RSC), is developed. This new model is similar in spirit to Constructive Solid Geometry (CSG), except that it is also capable of consistently modeling assemblies of real-world objects, and covers a wider class of realizable shapes.

Descriptive solid modeling schemes such as CSG or RSC, are not suitable user-level part definition models for integrated CAD/CAM systems. Following our CAM-oriented methodology, it is shown how an operational approach to mechanical part definition can serve as a foundation for a manufacturing-oriented integrated design language. Such a language must provide operations meaningful to both designer and manufacturer, resulting in part definitions expressed in terms of components with meaningful counterparts in production cycles. Reflecting this view of mechanical part design, a skeleton design language called DSG, based on RSC and suitable for a truly integrated CAD/CAM system, is proposed and its major features are discussed.

Foreword

This research is concerned with Computer Aided Design (CAD) and Computer Aided Manufacturing (CAM) of mechanical parts and with the benefits of an approach to three-dimensional solid definition that considers manufacturing processes during the design. This work began in the context of research on requirements and characteristics of an "ideal" language for Computer Aided Design. It was clear at the outset that the diversity of CAD applications (e.g. design of logic circuits, mechanical parts design, architectural design, software design, etc.) mandated focusing on a more specific goal than a generalized CAD language. We chose to concentrate our efforts on Computer Aided Design and Manufacturing of mechanical parts and initiated a study of different approaches to definition and modeling of solid objects. Chapter I begins by a brief overview of CAD/CAM and discusses some fundamental concepts and terminology used in this field.

Although the inadequacies of Computer Aided Drafting systems were apparent, we, nevertheless, studied drafting as an informal modeling scheme for representation of mechanical parts because of its historic significance and traditional success. The concern in "good drafting practices" for manufacturing-related issues, the rising demand for integration of CAD and CAM, together with the recent trends toward omnipotent Computer Aided Engineering systems and automated factories, soon led us to the conclusion that a language for CAD must be based on a view of design and manufacturing broader than just an abstract mathematical model for geometric shape composition. Such a language must make it possible for designers to sense and deal with manufacturing-related issues which directly affect cost, performance, functionality, and feasibility of their designs. This language must produce part definitions in a form which can be directly used for automated fabrication. Chapter II briefly sketches the requirements of an integrated CAD/CAM system for design and manufacturing of mechanical parts, based on such a manufacturing-oriented view of design.

Geometric shape composition is indeed a central concept in any CAD/CAM system that deals with solid objects. Several representation schemes and modeling formalisms have been proposed and are used in existing systems. Chapter III reviews most of such popular solid representation schemes and discusses their potentials as candidate models for representation of mechanical parts in integrated CAD/CAM systems. This chapter also includes an extended introduction to several experimental and commercially available systems which typify distinctive approaches to CAD/CAM. Chapter III indicates that geometric solid modeling, although far from being sufficient, is a fundamental requirement for systems aimed at design and manufacture of mechanical parts. This is evident both from recent directions in theoretical research and in new trends in development of such CAD/CAM systems.

The major contribution of this research is contained in Chapter IV. This chapter reflects an attempt to first establish a sufficiently general mathematical formalism for modeling of real-world solid objects, and then, to show how an operational approach to mechanical part design can serve as a foundation for a manufacturing-oriented integrated design language. The first section of Chapter IV presents a new geometric solid modeling scheme called Realizable Shape Calculus (RSC), which is mathematically expressive enough to uniquely and unambiguously represent any physically-realizable object. An interesting characteristic of this new mathematical model is its unique ability to consistently represent assemblies of objects positioned arbitrarily relative to one another. The remainder of this chapter justifies an operational, manufacturing-oriented approach to design and definition of mechanical parts. Reflecting this view of mechanical part design, a skeleton manufacturing-oriented design language called DSG, based on RSC and suitable for a truly integrated CAD/CAM system, is proposed and its major features are discussed. Finally, some interesting areas for further research are briefly mentioned in a discussion on several theoretically intriguing and practically important problems; these include modeling of shaping operations, tolerances, and forging envelopes. Possible directions to address some of these issues within the operational framework of DSG are also pointed out.

CHAPTER I

Computers in Design and Manufacturing

This work focuses on the Computer Aided Design and Computer Aided Manufacturing of mechanical parts and thus, involves concepts and methodologies related to all aspects of manufacturing production cycles. Traditionally, two major phases of design and manufacturing are identified and separated, sometimes to the limit of isolation, in the fabrication of mechanical parts. Despite the relatively high initial investment, use of computers in (semi)automation of activities in design and manufacturing, has separately proved to be a viable means to increased productivity and efficiency.

Traditional structure of manufacturing production cycles reflects requirements and limitations of conventional fabrication techniques. Historically, increased complexity of manufacturing processes on one hand and acceleration of technological advancements on the other, necessitated "specialization" which began to separate "design" from "manufacturing". Industrialization of (once truly "manual") manufacturing further magnified this separation, and the two tasks which used to be performed or supervised by the same "craftsmen" are now delegated to isolated ranks of people who, therefore, must communicate. Today's costly iterations between design and manufacturing phases of production cycles are a result of this (justified) separation and lack of better means of communication. Computerization of factories will eventually both provide a proper channel of communication between designers and manufacturers, and mandate a revision of their responsibilities due to its profound effects on the structure of production cycles.

CAD and CAM systems have evolved in the last decade from being regarded as expensive fancy toys to practical tools imposed even on reluctant segments of manufacturing industries by the force of competition for survival. Furthermore, the necessity for integration of CAD and CAM is acknowledged by both researchers in this field and industrial users of CAD/CAM systems. Yet, despite marketing claims of some CAD/CAM system vendors, no truly integrated CAD/CAM system currently exists for design and manufacturing of mechanical parts. Indeed, certain critical issues must still be further elaborated on by researchers and tested in experimental systems before such integrated systems can be put into production use.

This chapter presents a brief discussion of major terms and concepts related to use of computers in design and manufacturing, emphasizing applications that involve mechanical parts.

1 Introduction

Most literature on "CAD" concerns not computer aided design, but rather, computer aided drafting. Ironically, in its most general form, design may involve no drafting at all. Furthermore, much of this literature should be better categorized as computer graphics. This misuse of terms becomes worse when design objects are mechanical parts whose functional significance, unlike, for example, logic circuits or software, crucially depends on their geometric shapes. We feel it is important to emphasize clearly the differences among graphics, drafting, and design, since they are distinctly separate methodologies used in many diverse disciplines, sometimes with overlapping application domains.

Vision is a non-sequential wide-band channel of communication for human beings. Visual effects, thus, become a vital means of information exchange, especially when information's volume and complexity pass the threshold of intelligible sequential processing. Since design very often involves large volumes of information, graphics and drafting (as an informal graphic modeling scheme) play a central role in design. Graphic representation is a fundamental means of communication in the design to manufacture cycle of production. Therefore, it is only natural that the most visible (no pun intended) aspect of CAD/CAM systems is their graphic capabilities. However, geometry, the underlying foundation of graphics, plays a more fundamental role in design and manufacture of mechanical parts than as only a pictorial representation scheme. Geometry encompasses every aspect of fabrication of mechanical parts, from drafting, the traditional methodology for part design and definition, to modern manufacturing processes using computerized numerical control machines. The close tie of drafting with graphics, its mathematical basis of projective geometry, and its significant role in design, can lead to a confusion that, unfortunately, is intensified by misuse of terms both in literature and in the CAD/CAM industry.

Pictorial representation of design objects (or of symbols designating design entities) involves graphics, irrespective of the significance or nature of represented entities. Definition of a design, especially of a mechanical part, often closely follows traditional drafting practices, again, somewhat independent of the nature of the design entity. The role of geometry in formal models of design objects within a CAD system, however, depends on the nature of those objects. For example, when design objects are electronic elements or circuit boards, contribution of their geometric properties to their function is relatively insignificant. Consequently, in an abstract model for representation of such entities within a CAD system, geometry plays at most a minor role. Such a model primarily deals with concepts of voltages, currents, impedance, etc., and only secondarily may involve simple geometric or topological properties such as lengths or connections. On the other hand, in modeling of mechanical objects, often all relevant information is geometric in nature. As a result, in addition to its significant part in pictorial representation schemes, geometry plays the fundamental role in modeling formalisms used for Computer Aided Design of mechanical objects.

The view promoted in this dissertation regarding integration of CAD and CAM is that CAD should be oriented toward CAM. Consequently, we are proposing a manufacturing-oriented design methodology that leads to a representation model and a design language suitable for an integrated CAD/CAM system. We believe it is essential to understand and consider the impact of manufacturing realities on the design process of mechanical parts in order to lay the proper foundations for an integrated CAD/CAM system. In a truly integrated CAD/CAM system, part definitions should be in terms of components and primitives which have meaningful counterparts in manufacturing. This permits, or indeed leads, designers to understand and evaluate the implications of manufacturing processes on feasibility, functionality, and cost of their designs before they are released for fabrication. Consequently, more of the subtle problems that often become apparent only at manufacturing, can potentially be dealt with during design when consistent solutions (as opposed to often hasty last-minute patch-works) are still possible. We refer to this approach as "CAM-oriented CAD methodology" and will pursue it further in Chapter IV by proposing a new modeling scheme for design and definition of mechanical parts.

2 Graphics, Drafting, and Design

Graphics is the art of making drawings (as in architecture or engineering), producing graphic arts design, or using typography. In short, graphics is the art of producing visual effects; it is distinguished (in a fuzzy way) from other visual arts such as painting, photography, and (artistic) drawing through its closer tie with mathematics, although this sometimes means no more than use of geometric entities, abstractions, regularity, or uniformity.

Drafting is the act of producing engineering drawings, i.e. of representing objects graphically with mathematical precision. Although such graphic representation of an object cannot be a complete description of that object and, in itself, is not sufficient for all engineering purposes, drafting, nevertheless, is used as a major tool in every engineering discipline. Drafting is a subfield of graphics where pictorial communication is for the most part confined to the rigidity of mathematical transformations of projective geometry. The result of a drafting process is an engineering drawing, which basically consists of a geometric description of an object in the form of one or more planar projections; additional required information may still be represented graphically on the same drawing, as in architecture.

Design is the process of planning and delineating an object, a scheme, an action, a mechanism, or an abstraction, primarily by sketching (not necessarily literally) its outline. The designing of an object begins with a designer's initial conception and ends with a variety of documents defining the object. The input to this process is usually a somewhat vague specification of the object, but its output must be precise. Indeed, one may describe design as a non-linear programming problem wherein the designer attempts to construct an "object" or a "system" in such a way as to optimize some criterion function subject to certain constraints. Unfortunately, the criterion function and many constraints are often poorly defined, and thus the design effort involves both mathematical and heuristic approaches.

Organizations have traditionally stressed separation of the design process from subsequent activities, including feasibility studies and manufacturing, as reflected strongly in both departmental organizations and production cycles. Although there are sound historical reasons for persistence of this traditional separation, in practice, many iterations between design and subsequent processes are caused by designers' lack of understanding of constraints and activities involved in manufacture of their design objects.

A designer is and should be concerned primarily with function, form and fit. He[1] designs a new object which is to perform a given function; this is his most important goal. Since his design is not meant to remain an abstraction but to perform the given function, he must also be concerned with the form in which his design is to be materialized. A third limiting factor on a designer's creativity is imposed by the fact that the end product of his design should fit into its intended environment (both physically and socially).

In design of mechanical parts, function, form, and fit, all manifest themselves in terms of parameters affecting the real world of three dimensional physical objects. The mutual interdependence of function, form, and fit presents a puzzle to mechanical part designers who must simultaneously satisfy all three design criteria. This is in contrast to the case of software or electronic design where a design object's function, form, and fit represent three rather-loosely-dependent smaller problems which can be separated and solved

1 "He" and "his" are intended to be generic.

individually. Architectural design involves still more complexity because it includes additional concern for aesthetic values which are extremely difficult to quantify, and for the fit of design objects, particularly, into their intented social environments. This challenges architects into human factor issues that frequently outweigh other technical problems both in volume and complexity.

Strictly speaking, there is no mandatory link between design and drafting, much less between design and graphics. Drafting is simply one convenient means of communicating the output of a design process, particularly in the case of mechanical parts; it is a language in which much of the precise definition of a part can be expressed and documented in a way that is "easily" understood by (trained) human beings, such as manufacturing engineers. Without additional information, a drawing is not a complete description of an object. However, drafting is a convenient means of defining geometric properties of objects because it provides a "natural", concise, and structured description which is precise enough for mathematical deduction of certain latent information.

Nevertheless, the "naturalness" of the choice of drafting as the language in which the outcome of a design process is to be documented is in part due to a self-perpetuating system of training which has deep historical and social roots. Even if more appropriate means of part definition were introduced today (and they are emerging now, in fact), the conventional "blue prints on paper" will remain with us for quite some time, due in part to social issues like fulfillment of legal requirements, contractual agreements, and technical publications.

As in most other fields, computers have been introduced into graphics, drafting, design, and manufacturing to facilitate or automate traditional practices without any significant attempt at their integration. CAD and CAM as acronyms for Computer Aided Design and Computer Aided Manufacturing, have been misused in both literature and in the "market", just like other fashionable terms in their time. CAM, for example, quite often is equated with computer systems which help to generate Numerical Control (NC) tapes, although NC is only a small part of CAM and antedates both computers' use in graphics and the terms "CAD" and "CAM".

Once a problem is defined, a solution may be found for it, and if the solution method can be generalized to deal with a commonly-recurring class of similar problems, then the solution method can be made into a "tool". A tool, by its virtue, is capable of solving a specific problem regardless of the context in which the problem arises. Therefore, the criteria by which tools are classified and related to applications should be based not on usability of a certain tool in a given application (since any tool may find its way into a wide variety of unrelated applications), but rather on the magnitude of the direct contribution of a tool to the <u>main problems</u> in an application.

Computer systems too are problem solving tools and should be classified according to whether or not they contribute directly to the main task(s) of an application, as opposed to helping with (or automating) any number of subsidiary tasks. In this view, a computer drafting or graphic system with or without NC, can rightfully be promoted as a CAD/CAM system roughly to the same extent that a text processing system can be, since design and manufacturing only partially involve the tasks of graphic representation, drafting, NC tapes, and text processing.

3 Production Cycle

A complete manufacturing production cycle involves more than creation of a concise description of a product; in general, it starts with a crude informal specification which will then be refined into a part definition by a designer. This definition often involves engineering drawings and must go through an analysis phase to guarantee that it satisfies intended requirements and specification of the part. Part definition, drafting, and analysis characterize a typical conventional design process which may include several iterations among these subtasks before a part definition is finally released to subsequent phases in a production cycle.

The manufacturing phase of the production cycle follows the design phase and involves process planning, tool design, and NC tape preparation. Process planning consists of specification of a (desirably optimal) order of the individual steps and manufacturing processes required for fabrication of a part. Process plans are determined by manufacturing specialists according to various standards and on the bases of geometry, material, and tolerances of a part, within the range of capabilities of available manufacturing machines, tools, and operations.

Ideally, a computer provided with a sufficiently accurate definition of a part should be able to produce such manufacturing process plans. Use of computers in this capacity is called Computer Aided Process Planning (CAPP). To this end, a technique called Group Technology (GT) classifies objects into families of similar parts. In addition to CAPP, GT classifications have several important applications in design and manufacturing. For example, they permit a designer to search company records for parts similar to the one he contemplates to design; locating such parts provides him with valuable information that can lead to considerable time and cost savings in both design and manufacturing. Potentially, CAPP can benefit from GT classifications. Through associating generic process plans with families of parts (e.g. turned piece, sheet metal, etc.) and assigning codes to identify specific (orders of) manufacturing operations based on individual parts' geometry, the actual manufacturing process plan for a given part can be characterized by a finite-digit code. Despite availability of dozens of GT and CAPP codes, CAPP is still at its infancy and no fully automatic manufacturing process planning system is currently in use.

When a reasonable process plan for fabrication of a part cannot be determined (and this is often due to unavailability of a specific manufacturing tool or process, or to the unjustifiability of the use of such a tool or process), either special tools and processes must be designed to meet the requirements, or the part definition must be returned to the design phase for revision. Tool design and subsequent re-tooling of a shop is generally a costly effort which must be avoided unless very strongly justified. Preparation of NC tapes, which are programs for Numerically Controlled machine tools, can conceptually be viewed as a convenient and far less costly form of tool and process design.

Other tasks in a production cycle which may somewhat overlap design and manufacturing include assembly, testing and inspection, bill of material processing, inventory, and finally, technical publication.

4 Integrated CAD/CAM Systems

A CAD/CAM system must (at least) "aid" automation of two principal phases in a production cycle: design and manufacturing. People have spent much effort in "bridging the gap between CAD and CAM" by applying sometimes sophisticated heuristics to unveil "latent" information necessary for automation of production cycles. These approaches include, for example, attempts to automate two of the better understood tasks in manufacturing: process planning and NC. The implicit information sought by such heuristics is generally contained in object descriptions produced through a design process and, we believe, should be available in a directly usable form in the model of a truly integrated CAD/CAM system.

We believe that the best way to "bridge the gap between CAD and CAM" is to avoid creation of the gap in the first place. The nature of the problems involved in automation of a production cycle do not allow successful treatment of CAD and CAM in isolation. This view has very significant implications on the modeling of objects — an easy-to-generate, concise, complete, and unambiguous description of an object is no longer necessarily a suitable one for an integrated CAD/CAM system, because it may not be in a form directly usable for manufacturing. The basis of our proposal for integration of CAD and CAM is, therefore, a solid definition model for CAD/CAM systems that integrates the processes and constraints of manufacturing into the design and description of mechanical parts. In doing so, we feel that we also provide design engineers with primitive concepts that very closely relate to their conceptualization of design objects.

CHAPTER II

Requirements of a CAD/CAM System

This chapter sketches a brief outline of the major requirements and architectural components of an integrated CAD/CAM system. Of the various approaches to design and representation of three-dimensional solid objects, we believe solid modeling (see Chapter III) best fits the requirements of integrated CAD/CAM systems. However, solid modeling is by no means sufficient; it is only the proper embedding of a powerful solid modeler within an appropriately-structured, functionally-rich environment that qualifies as an integrated CAD/CAM system. Far from being a system design, this chapter merely depicts the functional environment within which we envision a suitable solid modeler to operate.

Modeling operations that a CAD/CAM system places at designers' disposal must be more structured and more sophisticated than elementary shape construction primitives of a simple solid modeler such as CSG (see Chapter III). The view thereby presented to designers by such more sophisticated operations in turn, constitutes another CAD modeling scheme for design and definition of solid objects. Operations supported through this user-level model must be meaningful to both designers and manufacturers. This results in part definitions which are in terms of components with meaningful counterparts within a production cycle. We refer to this methodology as CAM-oriented CAD and pursue it further in Chapter IV.

Rather than trying to establish the cliched "bridge between CAM and CAD", we believe our CAM-oriented approach to CAD offers significant benefits to the design-to-manufacture cycle of production.

1 Model and Model Database

At the heart of a Computer Aided Design system, there must be a conceptualization of what is being designed. This conceptualization can be formalized as an Abstract Data Type which defines both the underlying domain of "values" of the type (together with "literals" denoting a subset of these values) and the set of operations that can be performed on this domain. The model of a CAD system is such an ADT view of this conceptualization.

Although there may be a substantial amount of software between a designer and the implementation of the model of a CAD system, the model is what a designer sees. He perceives the model representation of a design object and expresses his actions in terms of the set of operations supported by the model, i.e. the modeling language. Clearly, the farther away the model is from "natural" concepts with which designers deal, the greater the effort required for perception and manipulation of the model. A closely related issue is how faithfully the model represents reality. The "accuracy" of a model may be measured by the amount of relevant information about a design object kept in the model, and also, the amount of information kept in the model that is directly relevant to the design process. Lack of sufficient relevant information in the model results in incomplete or ambiguous representations, or creates impossible objects which must be checked and possibly rejected after their creation. Any piece of information contained in a model should be of significance to its users. Existence of not-directly-relevant information in a model reveals a missing level of abstraction between the model and its users, i.e. unsuitability of the existing model for the existing application.

In order to model a subset of the real world properties of design objects so as to aid generation, deduction, correlation, and checking of detailed information describing design objects, a CAD/CAM system requires a suitable model and a data base for storage and retrieval of a representation of that model. Every other function in the system can be viewed as an independent application which uses and modifies the model database.

It is quite conceivable that such applications would choose to see the information content of the model database in different fashions according to their own requirements. This is analogous to a database shared by a number of different applications, where each must see a certain "view" of its information content. Similarly, a CAD/CAM system should support different views of its model for different applications. Furthermore, the model should be informationally complete with respect to requirements of all such applications. Among important applications in a CAD/CAM system are (graphic) representation of the model, a designer end-user interface, manufacturing process planning, and numerical control programming. Geometric properties of a mechanical part are of prime interest in all such applications.

The three-dimensional nature of mechanical parts cannot be conveyed unambiguously in a two-dimensional model. Of the various approaches to three-dimensional design, we feel solid modeling best fits our requirements. The amount of information kept in such a model allows efficient analysis and flexible display of properties of objects.

Nevertheless, we regard solid modeling as necessary but not sufficient for a truly integrated CAD/CAM system. The full set of activities within an integrated CAD/CAM system requires a model for *fabrication* of solids, not just *modeling* of solids. The most accurate model for a solid object, the object itself, is not necessarily the most appropriate form of its description for manufacturing purposes. The model database in an integrated CAD/CAM system must contain enough descriptive information about design objects to efficiently respond to queries involving their geometry, *and also* be capable of directly supporting manufacturing-related applications such as process planning.

2 End-user Interface

An end-user interface provides a means for expressing users' intentions in a form tailored to their needs, as determined by the nature of their applications and by their personal preferences. At the same time, it must fully utilize any specialized hardware and software features that may be available on users' work-stations to achieve their goals.

In principle, the modeling language should suffice for all users' needs at least as long as they are dealing with modeling activities. However, we regard the modeling language not so much as a user-oriented language, but primarily as a language in which users' manipulation of design models can be expressed unambiguously. Therefore, one of the goals of an end-user interface is to permit designers to perform their design and modeling activities in a convenient manner and through a friendly language. In addition, users of a CAD/CAM system must be given facilities for "housekeeping" tasks (such as accounting, maintenance and security of the model database), utilities (such as protocols for communication among users and between machines), and procedures for in-house standards by which they must communicate.

An ideal end-user interface takes advantage of resources and facilities available at individual work-stations to provide a pleasant and friendly environment for end-users. A work-station itself is more justifiable as user-community property (if not a personal one), rather than system property, and users rather than system designers should decide how their work-stations operate. They should also have the capability of re-tailoring them to their changing requirements. A CAD/CAM system designer influences the user-system communication through the structure of the modeling language, and by deciding what the overall system capabilities are and what the modeling language can do; the choice of how to perform modeling operations and how to use system capabilities should be left open for end-users.

Architectural isolation of work-stations from functionality of a system can also be justified from a system's point of view: with the ever increasing diversity of work-station terminals and advanced capabilities available through utilization of micro-processors, it is impractical for a CAD/CAM system designer to consider sufficient end-user related options, especially since his system design will likely outlast its initial hardware. Alternatively, it is quite advantageous for a system designer to define clearly what a CAD/CAM system can do in the most "device-independent" form possible. Structuring a system into several autonomous functional layers whose interdependence is limited to mutual communication at a functional level, makes the system easier to understand for both users and implementors. It also has the advantage of allowing both implementation flexibility and application versatility, especially in the part of the system where change is most likely and to which end-users are most exposed, i.e. the user interface. Extraneous enhancements, alternative shortcuts, and operational details pertinent to a particular piece of hardware, a user community, or some specific application must affect only the outermost architectural layer of such a system.

Available hardware and intelligence on work-stations should be utilized to minimize user intervention, and in particular, mechanical intervention. To this end, dynamic expansion of keyboard entries, menus, light-pens, styli, function keys, and voice recognition devices are but a few techniques that could be incorporated into a user-interface. It should be straightforward to tailor an interface to the specific needs of an application and to the special features of a work-station if (and only if) proper analysis is done by qualified and application-knowledgeable human factors personnel. This requirement has often been neglected or camouflaged by "bells and whistles".

On the other hand, lack of facilities on a less expensive work-station should not introduce an impossible-to-cross barrier to usefulness; it should rather mean a reduction of service due to the lack of special features, with absolute compatibility preserved at a functional level.

2.1 Modeling Language Interface

The purpose of a modeling language interface is to make model manipulation as convenient as possible for end users. Depending on the type of user, hardware available, and the nature of an application, a variety of dialects of a modeling languages may be supported on a CAD/CAM system. Generally, menus are preferable to procedural interfaces; however, they limit flexibility for more experienced users who would like to get more involved with the system. Informative messages, verbosity, and a limited set of "next action" choices are valuable means of assistance and education for a novice or an occasional user, whereas to a proficient user, they are useless wastes of time and bothersome road blocks.

Lengthy input sequences should be avoided or broken down into meaningful smaller ones. Also, the nature of the input parameters must be relevant to the application, i.e. users should be expected to supply information in forms convenient for them, rather than in forms directly usable in the model. It is the task of a modeling language interface to reformat, extract, and combine such user-oriented input to generate information required by a modeling operation.

To this end, utilization of graphic input is a major technique. However, graphic input must be differentiated from construction of a graphic model (as in drafting). In contrast to two-dimensional systems where a model is directly composed of graphic entities that users create, the role of graphic input in a system based on solid modeling is only to provide parameter values required by the modeling operations; with or without such graphic input capability, it is always the modeling operations that define and construct a model. For example, in a DSG-based system (see Chapter IV), the modeling language interface can provide users with a "transparent overlay" on which they can make fast working sketches similar to markups one would make on a workpiece. Then, for example, instead of giving parameters that define the position and orientation of a planar-cutter directly, a user can draw a line on such an overlay, position (turn and translate) the workspace to place the workpiece at the desired location and orientation, and then indicate that the line should be used as the source of the parameters defining an application of a planar-cutter. It is important to emphasize that such drawing sketches are not parts of a model, nor are they bound in any way to specific modeling operations; they are merely "graphic expressions", i.e. short-hand conventions, which are evaluated by a modeling language interface and permit a more natural means of supplying parameters to modeling operations.

The open-ended requirements alluded to above, sketch a boundary for a modeling language interface. A modeling language interface should not functionally expand or augment a modeling language; rather, it should utilize modeling language capabilities in a situation-dependent fashion in order to present a high level convenient set of operations to end users.

2.2 Model Representation

The other direction of the user-system communication involves model representation. The task of the system component which handles model representation is to determine how a design model and any additional non-graphical information can be presented at designers' work-stations. Within the constraints of available hardware, users should generally be able to define multiple viewports and to dynamically allocate and release areas of their graphic screens to viewports. The content of each viewport should also be user-definable, e.g. orthometric, isometric, or perspective views of the design object, with or without information like hidden lines, dimensions and tolerances, menus, messages, context, etc.

2.3 User-System Interaction

The interface between users and the system as a whole, must meld quite well with the modeling language interface so that users are presented with one homogeneous communication language throughout interactive sessions and not with different sets of communication standards. Among the responsibilities of the user-system interaction interface are logon/logoff procedures; accounting, filing, and archiving; application selection and execution; and model data base security and access provisions.

CHAPTER III

Approaches to CAD/CAM

This chapter presents an overview of the field of mechanical CAD/CAM, covering both its theoretical bases of modeling and its present practical realities materialized in the form of today's available commercial and experimental systems. The non-balanced ratio of topics involving CAD to those on CAM in this chapter, is in fact a representative picture of the unbalanced scale of work done on CAD compared to that devoted to CAM, both in theory and in practice.

Section III.1 is a study of the more traditional representation techniques for physical objects, scrutinizing them as candidate modeling schemes for an integrated CAD/CAM system. Section III.2 is a natural continuation of Section III.1 and examines other solid representation methods. Separation of the material in Sections III.1 and III.2 reflects the fact that the methods of Section III.2 are more recent, both in their development and in their being used as representation schemes in CAD/CAM systems. The significant difference between the two classes of methods, however, is that modeling schemes of Section III.2 are unambiguous, whereas representation techniques of Section III.1 are inherently ambiguous models of solid objects.

Section III.3 sketches a brief overview of available systems and their capabilities by reviewing a number of "typical" CAD/CAM systems aimed at mechanical applications.

1 Representations of Solid Objects

This section considers the more traditional approaches to representation of solid objects. Engineering drawing, the workhorse of communication between designers and manufacturers for the past centuries, and its obvious extension to three dimensions, the so-called wire-frame model, comprise the subject of the present section. In addition to popularity, these methods share another characteristic: their ambiguity. In the case of engineering drawings, their ambiguity imputes to them a degree of robustness that allows them to capture more than just the representational aspects of mechanical parts, to the extent of becoming models for manufacturing as well as for design. Although drafting is too informal a model to be useful as the backbone of integrated CAD/CAM systems of the future, it, nevertheless, is capable of addressing certain manufacturing related issues far better than other models currently do. An examination of drafting, then, is beneficial for sketching a perspective within which some of the problems that must be resolved in an integrated CAD/CAM system can be identified.

1.1 Engineering Drawings

The primary aim of an engineering drawing, or indeed of any other type of definition of an object being designed, is to provide the information required to manufacture that object. A drawing is mostly an indirect description of "what" a designer intends to have built (as opposed to a description of "how" it can be built) which, due to the lack of better means of communication, is expressed in the form of a collection of projections. Strictly speaking, such a collection does constitute a model of an object. However, in general, this "model" is too informal and ambiguous, and several serious problems arise when one attempts to use engineering drawings as the main form in which object descriptions are kept, particularly, in an automated system. Traditionally, production cycles have relied on the "common sense" and intelligence of trained human beings to resolve these problems. Our technological achievements serve as the proof that this arrangement often works successfully; the mere existence of the rank of drawing checkers in traditional production cycles, and, still, the costly iterations between design and manufacturing show that this means of communication sometimes fails. An automated system cannot rely on "common sense"; part descriptions in such a system should be in a form suitable for straight-forward formal deduction of properties which are of interest for its fabrication, and engineering drawings cannot fulfill this requirement for a variety of reasons, as we will show in following sections.

1.1.1 Projective Geometry and Drafting

Drafting is based on projective geometry, which deals not with attributes and properties of an object (i.e., metric and topological relations between geometric entities comprising the object's shape) directly, but rather with what becomes of such attributes and properties after they go through a process called projection. Engineering applications commonly use planar projections, i.e. projections of objects onto planar surfaces. Each such projection is called a view, and an engineering drawing consists, essentially, of a number of views.

Frequently, more than one view is required to give a "reasonable" description of an object because a great deal of information is (intentionally) suppressed through every projection. Drafting text books state that a sufficient number of views, section views, details, and notes should be given in an engineering drawing to clearly define the represented object. In drafting, each view in turn must be mundanely defined in terms of elementary geometric entities, e.g. points, lines, and curves. There are numerous types of planar projections, each more suitable for certain purposes than others, and some drafting texts include extensive surveys of such techniques [BOO63, LUZ59, FRE66, ARB81, MEL82b]. Not every set of lines and curves, however, is a legitimate view of a real object, and it is not always easy to check whether such a set is a valid view. Figure III.1 shows an interesting and well known example of an invalid view (in the sense that it cannot be that of a real solid object).

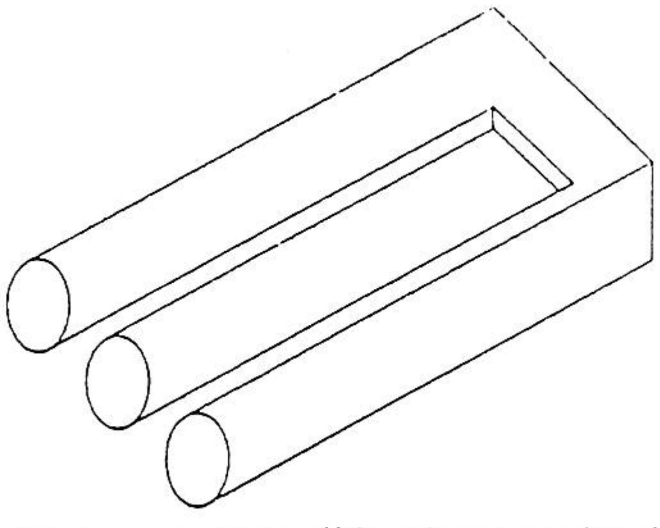


Figure III.1 - An Impossible Planar Projection View

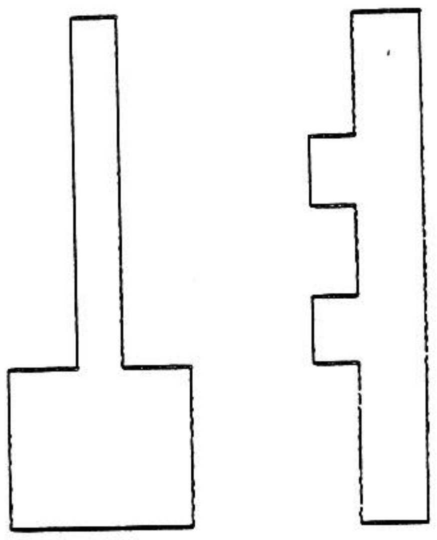


Figure III.2 - Inconsistent Views

There is a certain amount of redundancy of information in any engineering drawing because no two projection views of a three-dimensional object are absolutely independent. This redundancy serves to maintain some (complex) inter-view consistency relationship and implies that not every set of legitimate views can define a real solid, as seen in Figure III.2. Another problem with projective geometry is that objects which are smoothly curved in two directions, e.g. a sphere, cannot be defined precisely by any finite number of projections, although exact definitions of such objects can be approximated to any desired precision by large numbers of projection views.

Some of the problems in working with engineering drawings arise from the issues discussed above. It should be noted, however, that drafting is not exactly the same as projective geometry. Drafting practices have evolved through the years as a means of communication between designers and manufacturers and at times defy the rules of projective geometry. For the sake of brevity and convenience, drafting standards sometimes decree "erroneous" and/or "conflicting" views, not

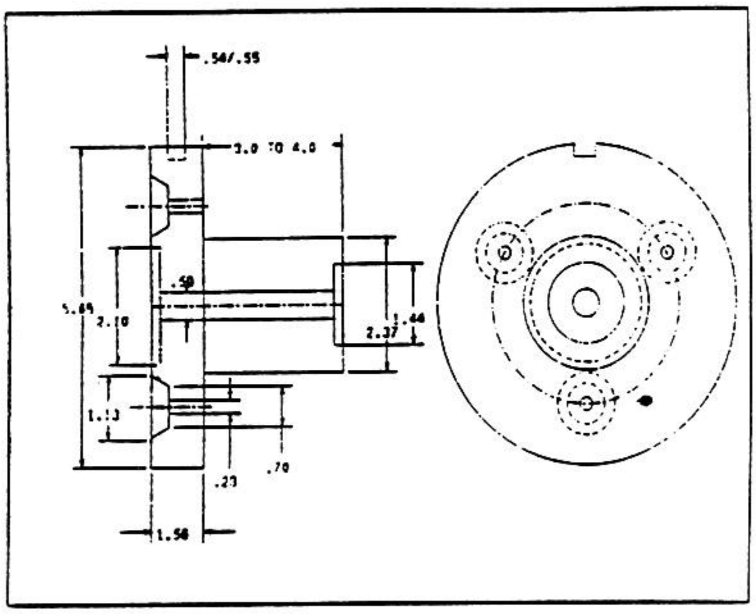
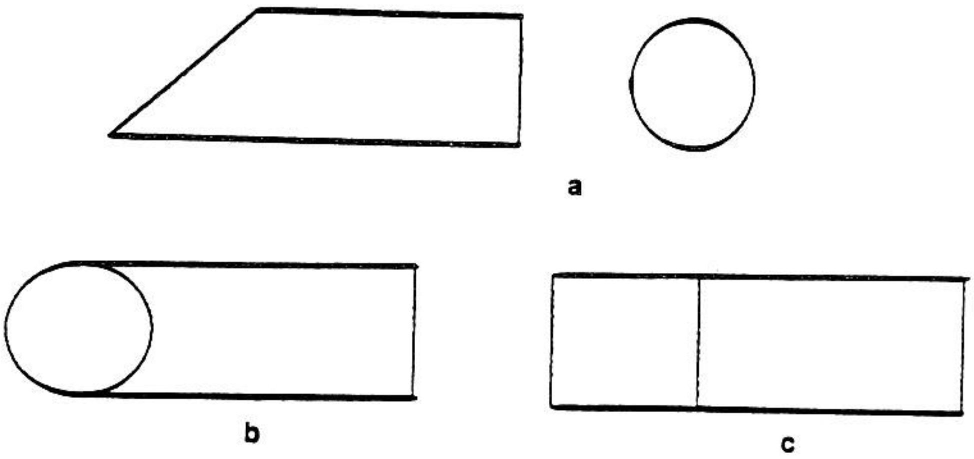


Figure III.3 - Misrepresented Holes

to mention suppression of information that can be inferred "obviously", or that can be deduced from notes and text. Whenever clarity may be enhanced through such measures, it is considered good drafting practice to apply them. In particular, the following examples illustrate why it is so difficult to extract information from an engineering drawing in an algorithmic fashion. Figure III.3 shows the front and side views of a turned piece. Notice that to highlight the part's axial symmetry, the positions of the two top off-center holes in these views are misrepresented. Figure III.4.a shows two views of a simple cylindrical rod beveled at one end. Figure III.4.b shows the "true" top view of this object as per rules of projective geometry. Figure III.4.c shows the top view of the rod as represented in a real-life engineering drawing and is another example of what is known as "preferred-" or "conventional-views". These intentional "errors" are somewhat typical of errors sanctioned by some official drafting practices standards and what is referred to as "conventionalization" (of views).



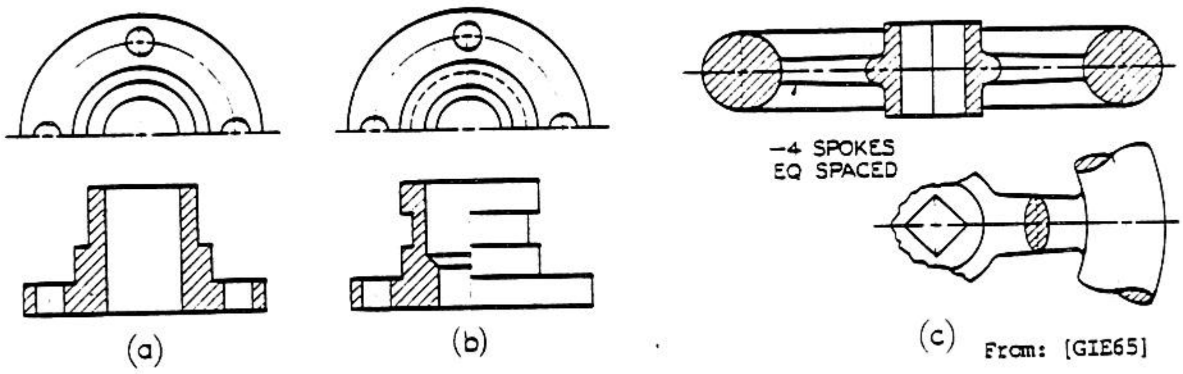
(a) Front and side views of a beveled rod

(b) True top view

(c) Conventionalized top view

Figure III.4 - Misrepresented Top View

Partial views, Figure III.5, representing only the relevant parts of a view, and about a dozen different types of section views (e.g. full, half, broken-out, revolved, removed, offset, phantom, etc.) are other points of departure from projective geometry where drafting adopts more informal schemes for the sake of clarity.



From: [GIE65]

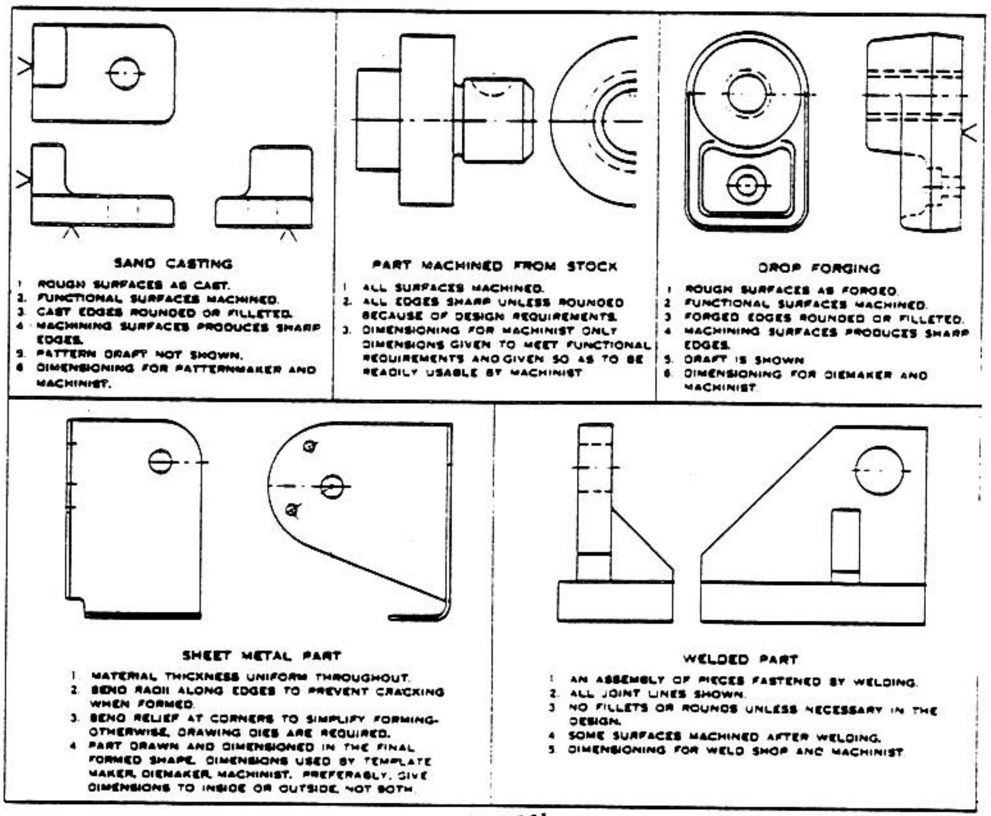
Figure III.5 - Partial Views

1.1.2 Manufacturing Issues in Drafting

An important aspect of drafting is that an engineering drawing is more than a representation of an object; it includes numerous hints for manufacturing of the part it describes and thus can be regarded as an informal (ambiguous and incomplete) model for manufacturing of the object it represents. Drafting text books emphasize that manufacturing processes involved in making of a part should be of prime concern in drafting of its engineering drawings. Certain aspects of a drawing, such as representation of detailed features of a part and, to some extent, the choice of dimensions, depend on manufacturing considerations.

Many drafting text books include an entire chapter on "shop processes", almost invariably, immediately preceding the chapters on dimensioning and tolerancing. Presented material under this topic covers techniques such as pattern making, sand casting, forging, lathing, milling, grinding, shaping, honing, and broaching, and the effect of these basic manufacturing methods on engineering drawings. Manufacturing processes are commonly classified into five categories for draftsmen's purposes [FRE58, FRE66, and GIE67]. Drawing requirements for each category are somewhat different as can be seen in a summary in [FRE58 and FRE66] reproduced here as Figure III.6.

Dimensioning standards differ somewhat depending on the nature of a part, manufacturing processes involved, and organizations' preferences. However, every drafting practitioner would agree that the prime purpose of dimensioning is <u>not</u> to define the part represented in a drawing



SAND CASTING

1 ROUGH SURFACES AS CAST.
2. FUNCTIONAL SURFACES MACHINED.
3 CAST EDGES ROUNDED OR FILLETED.
4 MACHINING SURFACES PRODUCES SHARP EDGES.
5. PATTERN DRAFT NOT SHOWN.
6 DIMENSIONING FOR PATTERNMAKER AND MACHINIST.

PART MACHINED FROM STOCK

1 ALL SURFACES MACHINED.
2. ALL EDGES SHARP UNLESS ROUNDED BECAUSE OF DESIGN REQUIREMENTS.
3. DIMENSIONING FOR MACHINIST ONLY DIMENSIONS GIVEN TO MEET FUNCTIONAL REQUIREMENTS AND GIVEN SO AS TO BE READILY USABLE BY MACHINIST

DROP FORGING

1 ROUGH SURFACES AS FORGED.
2 FUNCTIONAL SURFACES MACHINED.
3 FORGED EDGES ROUNDED OR FILLETED.
4 MACHINING SURFACES PRODUCES SHARP EDGES.
5. DRAFT IS SHOWN
6. DIMENSIONING FOR DIEMAKER AND MACHINIST.

SHEET METAL PART

1. MATERIAL THICKNESS UNIFORM THROUGHOUT.
2. BEND RADII ALONG EDGES TO PREVENT CRACKING WHEN FORMED.
3. BEND RELIEF AT CORNERS TO SIMPLIFY FORMING- OTHERWISE, DRAWING DIES ARE REQUIRED.
4 PART DRAWN AND DIMENSIONED IN THE FINAL FORMED SHAPE. DIMENSIONS USED BY TEMPLATE MAKER, DIEMAKER, MACHINIST. PREFERABLY, GIVE DIMENSIONS TO INSIDE OR OUTSIDE, NOT BOTH.

WELDED PART

1 AN ASSEMBLY OF PIECES FASTENED BY WELDING.
2. ALL JOINT LINES SHOWN.
3 NO FILLETS OR ROUNDS UNLESS NECESSARY IN THE DESIGN.
4 SOME SURFACES MACHINED AFTER WELDING.
5. DIMENSIONING FOR WELD SHOP AND MACHINIST.

From: [FRE66]

Figure III.6 - Manufacturing Issues in Drafting

mathematically, but rather, to aid manufacturers in fabrication of the part. To illustrate the difference, consider Figure III.7. Figures III.7.a and III.7.b show a single shape dimensioned in two different ways. Both dimensioning schemes are mathematically complete in that they present enough information for deduction of any desired metric property of the shape. Most practitioners would, however, consider the drawing in Figure III.7.a as poorly dimensioned since the one in Figure III.7.b is more directly useful to a manufacturer of the part.

General rules for proper dimensioning state that each dimension should be given clearly and in only one way, i.e. no duplicate or redundant dimensions; no dimension should be given except those needed for fabrication, inspection, and assembly of parts; dimensions should be

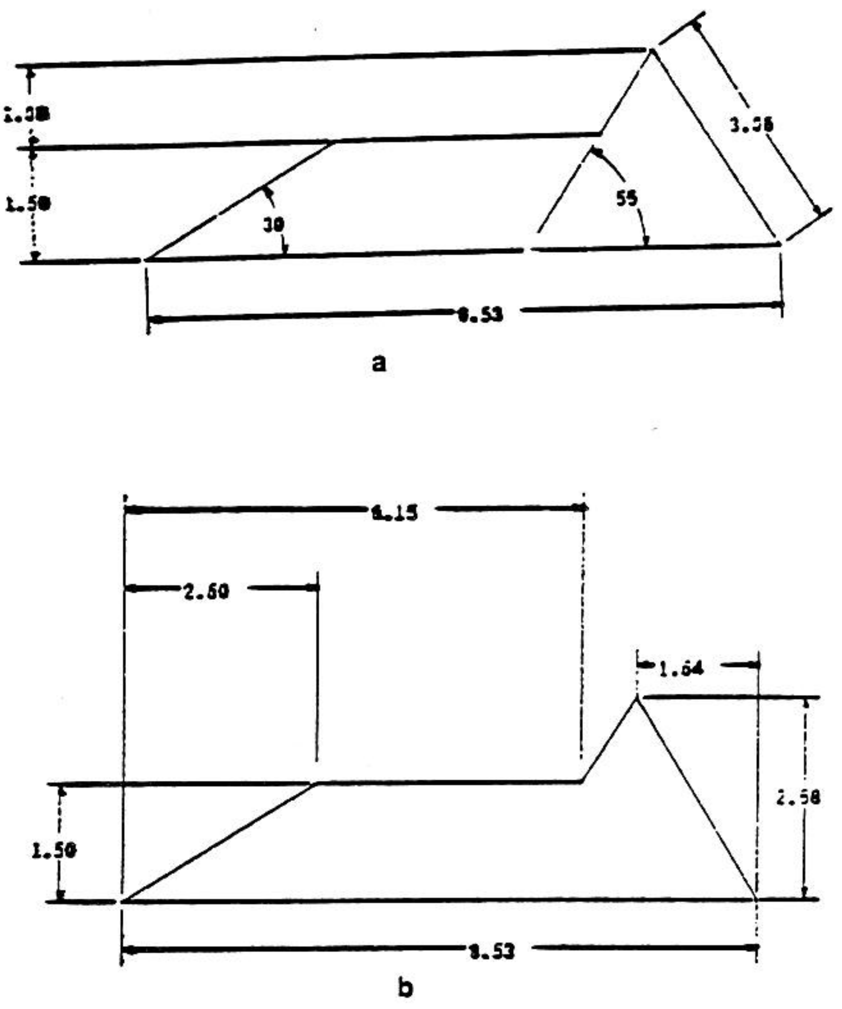


Figure III.7 - Proper and Improper Dimensioning

given between points or surfaces which have a <u>functional</u> relation to each other; dimensions should be attached to the view where the shape is best shown in true (scaled) size and, finally, it should not be necessary for manufacturers to calculate, assume, or measure a dimension.

1.1.3 Deciphering a Drawing

The fact that an engineering drawing represents certain geometric properties of an object after they have gone through the process of projection means that a draftsman or a designer must be keenly aware of projections and, in fact, must always perform this process on his conceptual image of an object before expressing it in a drawing. "Understanding" a drawing also, at least partially, involves the process of projection, or in fact, its inverse. Inverse projection, i.e. finding the geometric entity whose projections yields a set of given views, is far more difficult than projection itself. Yet, people who deal with engineering drawings, somehow manage to, in effect, solve this problem and "understand" drawings, even in spite of certain possible errors and inconsistencies. It is precisely the task of drawing checkers to catch such errors and inconsistencies in drawings and also to check for certain "manufacturability" criteria, all of which require a thorough understanding of the "meaning" of a drawing. In a computerized drafting system, it is quite reasonable to expect the computer to play the role of a drawing checker. In its most general form, however, such a task is an extremely complex one and involves sophisticated heuristics and pattern recognition techniques and falls into the mysterious realm of artificial intelligence.

We mentioned earlier that certain objects cannot be precisely (i.e. unambiguously) defined by any finite number of projections. Furthermore, it is not known whether reasonably general classes of (curved) objects can be represented unambiguously by a finite set of projections [REQ80b]. We do not know of any results on reconstruction of an object (or the set of all possible objects) from a given set of projections, in the general case. Several attempts, however, have been made to extract from a drawing certain properties of interest about an object, or to reconstruct it from engineering drawing type representations, under some special case assumptions. For example, it is known to be possible to unambiguously construct the set of edges of any planar-faced polyhedron from a sufficient number of its planar projections. Together with the hidden-line information contained in an engineering drawing representation in the form of dashed lines, this can yield an unambiguous reconstruction of such a polyhedron. In [PRE81], a set of algorithms for construction of a planar-faced polyhedron from its planar projections is presented. In [WES81] an earlier work on construction of polyhedral solids from their wire-frame representations [MAR80] is extended to derive solid models from projections. This interesting algorithm works on projections of (planar-face) polyhedra and accepts cross-sections, overall, and detailed views subject to some set of "standard" drafting conventions. In an attempt towards automated manufacturing process planning, the CAD/CAM Group at UCLA has developed algorithms for analysis of engineering drawings to extract certain shape characteristics that are significant factors in determination of a part's process plan [KAM83]. Subject to a relatively small set of not-too-restrictive assumptions, these algorithms in effect "understand" drawings of mechanical parts and automatically produce their CAPP codes (see Chapter I). Imposition of "standards" and structure on engineering drawings, which can be validated as a drawing is being constructed, can make computerized understanding of engineering drawings practical for real-life special, but important, cases. However, insuring validity of a drafting-style definition of an object in general, seems to be as complex as other poorly-understood cognitive tasks performed somewhat casually by human beings, and, at least for the near future, remains unsolvable.

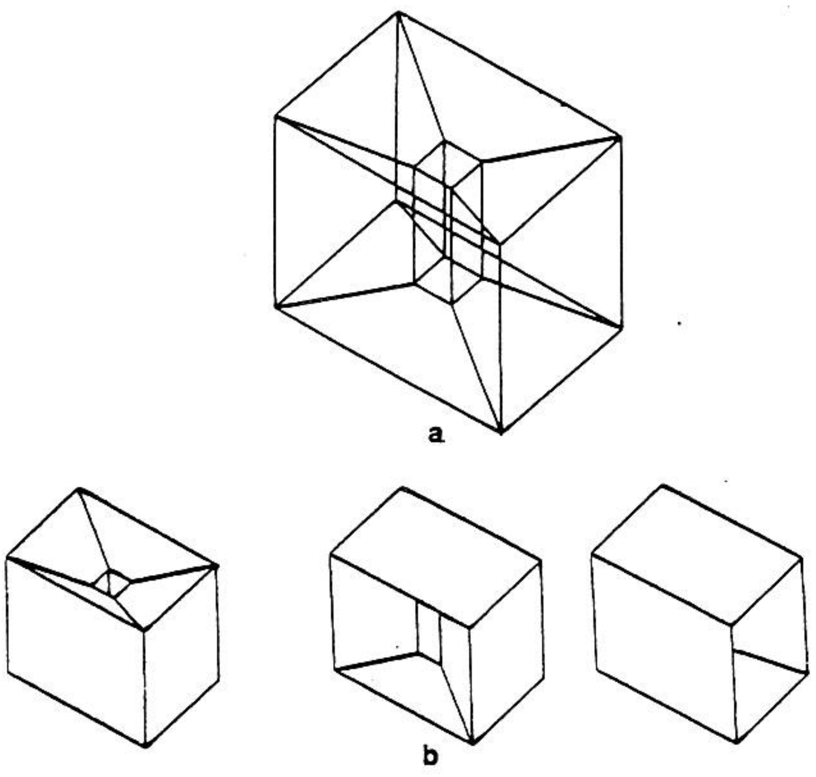
A more promising approach to construction of valid representations of solid objects is to define objects for the computer in a more direct fashion, i.e. abandon drafting as the language of communication between the designer and the computer. This approach has the additional benefit of alleviating the need for designers to constantly perform projections before they can express themselves.[1] Once an object is defined unambiguously in a computer system, the system can perform a wide range of tasks involving that object. For example, the main objective in using such a system may still be production of engineering drawings, in which case the system can perform the routine task of projection on a fully-defined object and produce any number of desired views in a straight-forward manner. Use of computers in such a capacity requires an "understanding" of the design object on the part of the computer beyond a mere superficial representation such as a collection of lines and curves, that bear an almost symbolic resemblance to the design object, expressed through the riddle of drafting. The collection of information comprising such an understanding is called a <u>model</u> of <u>solid</u> objects. Several approaches to geometric solid modeling have been introduced, and Section III.2 presents a brief review of this subject.

1 It also has the disadvantage of requiring people who have developed a habit of constantly performing projections, to stop and rethink!

1.2 Wire-Frames

A direct adoption of the concept of "primacy of edges", so fundamental in drafting, to three-dimensional space leads to the so-called wire-frame model of solids. A wire-frame model consists of a set of space curves (wires), each of which represents a "hard edge" of the solid object, i.e. a curve on the boundary of the object along which the derivative of the surface is discontinuous. Wire-frames are easy to generate and work with, are simple to store and manipulate in computers, and are sometimes very useful as visual aids. However, they do not capture enough of the shape properties of physical objects to convey the notion of solidity. This gives rise to a number of problems when wire-frames are used as models for solids.

In a wire-frame model, one can find no hint of any kind of a surface; surfaces are undefined entities whose intersections form edges which are explicitly represented as space curves. The shapes of surfaces whose boundary edges are represented by such curves, however, are left to be determined by observers' imagination. An engineering drawing not only represents every hard edge of an object but also shows a "silhouette" of the object in every view, often providing quite helpful hints to the shape of an object's bounding surfaces. Without the concept of projection, "silhouettes" are not possible and thus, in general, wire-frames are even less informative than engineering drawings in describing objects' bounding surfaces. In particular, curved surfaces cannot be dealt with in a pure wire-frame model. For example, the only "wires" that one can associate with a finite circular cylinder are two parallel circles; this is also the same representation as that of, among other things, two hemispheres. In cases of "very smoothly curved" objects, such as spheres, there are no edges to represent at all. As a remedy, in practice, people add wires where there are no real edges in objects, in order to approximate the shape of curved surfaces; these unreal edges are sometimes referred to as soft or fake edges and may be thought of as contours resulting from hypothetical planar sections of an object.



(a) Wire-frame

(b) Corresponding solids

Figure III.8 - An Ambiguous Wire-Frame

Another consequence of absence of surfaces in wire-frame models is that, in general, the notions of "inside" and "outside" are lost. Even when the shapes of surfaces are known (e.g. by convention or by approximation to planes), certain wire-frame models are still ambiguous, i.e. the same representation corresponds to more than one solid object. A classic example is shown in Figure III.8.a. Assuming that every face of the object represented by this wire-frame is planar, one can associate every one of the solids depicted in Figure III.8.b with this representation. (There are infinitely many more possibilities without the planar-face assumption.) In [MAR80] an interesting algorithm is reported which finds all planar-faced polyhedra that have a given wire-frame representation. Figure III.8.b was in fact produced by running an unreleased implementation of this algorithm under GDP/GRIN on the wire-frame of Figure III.8.a.

Absence of surfaces also makes it impossible (in fact, irrelevant) to talk about hidden lines. This, except for relatively simple objects, makes wire-frame representations too cluttered to be an effective visual aid, and the additional burden of including soft edges to represent curved surfaces only magnifies the problem.

Incompleteness of wire-frame representations limits their value for most CAM purposes. Wire-frame models are often augmented by (relatively simple) surface definition facilities to provide a more "complete" geometric model. With or without this augmentation, wire-frames are frequently used purely as a CAD tool, e.g. a faster means (than drafting) to generate correct and consistent projection views of an object. There are also certain applications, such as (semi)automatic generation of finite element models, where wire-frames provide reasonably sufficient representations.

2 Geometric Solid Modeling Schemes

The term "solid modeling" refers to a class of geometric models that unambiguously represent the shape of solid objects. The keyword "unambiguously" separates solid modeling from other representation methods such as engineering drawings and wire-frames. Many different schemes have been devised to represent and model real solids. In [REQ80b, REQ80c, and REQ82] extensive reviews of most such models and approaches are presented and [HIL82b] proposes a taxonomy of solid modeling systems according to their internal representation schemes and data structures. In this section we first briefly classify solid modeling techniques according to the above-mentioned references and then focus on several successful and well-developed representation schemes.

Briefly, [REQ80b] distinguishes between seven basic methods for construction of unambiguous representations of solid objects:

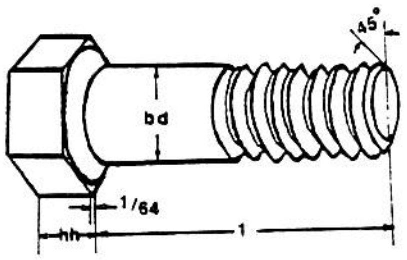
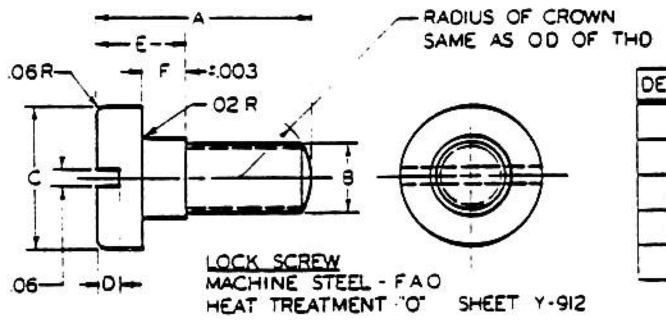


Figure III.9 - Family of Parts

1. Primitive Instancing

Primitive instancing is based on the notion of families of objects. Each member of a family is distinguished by a fixed number of predefined parameters and is called a primitive instance of the generic primitive represented by the family. For example a family of bolts can be represented by a tuple <BOLT,bd,hh,l> where BOLT is a type identifier and bd, hh, and l are body diameter, height of head, and length, respectively, as in Figure III.9.



DETAIL	A	B	C	D	E	F	UNC THD	STOCK	LBS
1	.62	.38	.62	.06	.25	.135	$\frac{5}{16}$ - 18	$\frac{3}{4}$ DIA	.09
2	.88	.38	.62	.09	.38	.197	$\frac{5}{16}$ - 18	$\frac{3}{4}$ DIA	.12
3	1.00	.44	.75	.12	.38	.197	$\frac{3}{8}$ - 16	$\frac{7}{8}$ DIA	.19
4	1.25	.50	.88	.12	.50	.260	$\frac{7}{16}$ - 14	1" DIA	.30
5	1.50	.56	1.00	.16	.62	.323	$\frac{1}{2}$ - 13	$1\frac{1}{8}$ DIA	.46

From: [FRE66]

Figure III.10 - Tabular Dimensioning

Primitive instancing has been used in the context of Group Technology in manufacturing and is the same concept used in many catalogs, technical references, and handbooks to define standard parts using a single drawing and tabular dimensions, as in Figure III.10. Primitive instancing presents an important abstraction scheme which is vital in many CAD/CAM applications, e.g. standardization of parts, material planning, and parts and material processing. The same conceptual view as presented by primitive instancing (family of parts) can be supported in other more structured approaches to solid modeling in the form of "macro definitions".

2.Spatial Occupancy Enumeration

In this technique, space is divided into a grid of three-dimensional cells (usually cubes), and a solid is represented by a list of the cells which it occupies. Because relatively accurate representations in this scheme would require a fine grid and this in turn results in bulky lists of cells, spatial occupancy enumeration is not used in mechanical CAD/CAM other than, as a possible internal "trick" to enhance performance of certain geometric algorithms.

3. Cell Decomposition

In this method, a solid is decomposed into simple cells with no holes and whose interiors are pairwise disjoint. Cell decomposition can be regarded as a generalization of the spatial occupancy approach where location, size, and shape of the cells are not fixed or prespecified. Cell decomposition is used in certain geometric algorithms which, for example, check whether a solid object is a single connected piece, etc. [REQ77c]. Cell decomposition is also important as the basis of finite element analysis methods.

4. Constructive Solid Geometry

In Constructive Solid Geometry (CSG), solids are defined as combinations of "solid building blocks" called primitive solids, through operations similar to volume addition and subtraction.

(See Section III.2.2.)

5. Sweeping

The basis of sweeping is that a solid or a bounded surface moving through space along a trajectory curve sweeps a "volume". A solid may therefore be represented by a pair <moving-body, trajectory>. (See Section III.2.3.)

6. Interpolation

Interpolation is a generalization of "lofting" refers to a technique used for approximation of surfaces by a set of longitudinal contour curves. Lofting is used mainly in applications where the surface primarily stretches in one direction, e.g. representation of an aircraft fuselage, ship hull, etc. Generalization of lofting to description of three-dimensional solids, involves defining a solid as the union of all lines $\overline{pq}$ whose end points p and q lie in two given two-dimensional sets P and Q, respectively. Interpolation is seldom studied or used in CAD.

7. Boundary Representation

In this method, a solid is represented indirectly by explicit representation of its topological boundary, i.e. its bounding surfaces. (See Section III.2.1.)

In addition to approaches mentioned above, it is possible to design solid representation schemes using a combination of these methods. Synthavision [GOL79], GLIDE [EAS80], GMSOLID [BOY80, BOY82], BUILD-2, DESIGN, and ROMULUS [HIL82a] are examples of systems using such hybrid schemes. Some of these systems, e.g. GMSOLID and DESIGN, actually use a hybrid (and redundant) scheme for representation of modeled objects [BOY79b, HIL82a]. Other systems in fact use a single representation scheme for modeling of objects internally, but support other modeling schemes at their user interface level as well.

The following sections further discuss boundary representation, CSG, and sweeping approaches to solid modeling.

2.1 <u>Boundary Representation</u>

Boundary representation is a method for describing a solid object in terms of its topological boundary. This boundary is itself divided into a finite number of faces or surface patches, each of which in turn can be defined and represented in a variety of ways. One popular method, for example, is to represent each face in terms of its boundary edges. Figure III.11 shows an object and its boundary representation definition.

Because boundary surfaces are really what one "sees" of an object, boundary representation seems to be a reasonable approach to definition of solid objects. Usually, it is intuitively clear to a designer what faces comprise an object's boundary, especially when decomposing planar-faced objects, where almost everyone would come up with the same set of faces. A succinct definition of a "face", however, is necessary

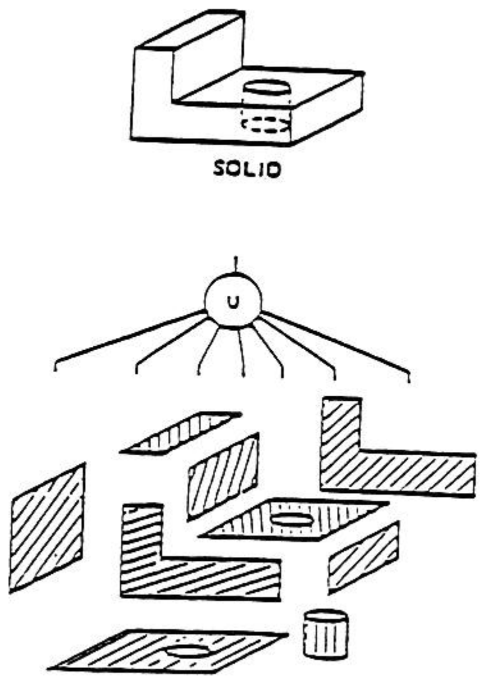


Figure III.11 - Boundary Representation of an Object

in the general case. Essential conditions that should be satisfied by a face are formally defined, e.g. in [REQ80b], and a number of definitions for various types of faces, leading to different decompositions, are presented in [SIL81]. The latter reference also contains a discussion of the mathematical properties of these faces and their associated boundary representation schemes. Furthermore, it discusses algorithms for conversion between representation schemes associated with different types of face definitions. Nevertheless, there are no definitely "correct" set of rules for decomposition of an object's boundary into a set of faces, and the choice of a particular decomposition is in fact application-dependent. For example, consider the object in Figure III.12. The division of this object's boundary into faces around the locality where the block and the cylinder meet is, in a sense, a matter

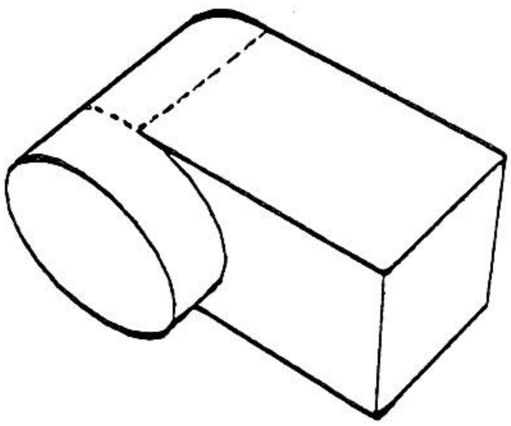


Figure III.12 - Ambiguous Faces

of taste. (Dashed lines in Figure III.12, for example, show two "natural" borders that may be considered as the separating edges of some such arbitrary faces.) Sometimes additional considerations such as primitive surface types available on a system, relative sizes of components, manufacturing processes, or a part's function, serve as hints at better or best face decompositions. But, since such issues are generally foreign to the model itself, they cannot, in general, be dealt with in a uniform manner.

Ready availability of definitions for surfaces, edges, and relations among them makes boundary representation a very desirable modeling scheme for graphic representation [GIL78, NEW79, LAN80]. For this reason, boundary representation is sometimes used as an internal intermediate form for generation of drawings on graphic displays, although the geometric model presented to users of the system may not be the same. GMSOLID, for example, keeps a boundary representation model of design objects in parallel to its primary CSG model, because the former is a more efficient representation for certain tasks including, in particular, graphic display of design objects [BOY79b, BOY80, BOY82].

From a designer's point of view, since in this scheme primary attention is placed on surfaces, which are higher level geometric entities than curves, boundary representation is a higher level method of definition than wire-frames. Unfortunately, however, definition of an object through its boundary faces is still rather bulky and often requires dealing in great detail with space points and curves in order to define a face or a surface patch. Depending on a particular system's user interface, this can at times become quite tedious and involved.

Definition and representation of surfaces (and their two-dimensional counterparts, curves) is in itself a specialized field of analytic geometry and graphics with a wide range of applications. It is instructive to consider three classes of applications discussed in [BEZ82] where "free formed" curves and surfaces are required. First, there is the class of applications where shape plays a basic <u>functional</u> role in some technical phenomenon whereby its characteristics are determined by scientific theories and engineering experiments. Examples include many engineering curve fitting applications, modeling of chromosome shapes [MER82], design of the shapes of propellers, ship hulls, aircraft wings, turbine foils, engine manifolds, etc. Second, there is a class of applications where the most important property of a shape is its <u>aesthetic</u> value, as in the "skins" of car bodies, furniture, household appliances, etc. Finally, a third class of applications deals with shapes which are not to be seen and therefore require no aesthetic properties, and further, need to satisfy only a "loose" set of functional requirements, such as not colliding with a moving part. The significant characteristic of this class of shapes is that they should be easy to fabricate. Examples include inner panels of car bodies, etc. These distinct classes of applications give rise to different curve and surface definition techniques.

Curve and surface representation techniques can be grouped into the two general categories of interpolation and approximation. Interpolation methods are usually used in the first and third classes of applications, above, whereas approximation techniques are mainly applied where aesthetics is the dominant factor. Specialized methods have been developed for definition of so-called "sculptured surfaces", a term that (usually) refers to doubly-curved surface patches, emphasizing the different characteristics of the final shape [FAU79, BAR82, GIL78, NEW79, FOR72, FEN80, GOR74a, GOR74b]. Most of these techniques use a definition of the boundary of a patch (usually, as a set of four, rarely three, connected space curves) or an equivalent set of constraints, together with a surface interpolant function that determines how such a frame is "fleshed out" into a surface. In the case of interpolation methods such as in Ferguson-Coons patches [FAU79, GIL78], this interpolant function guarantees that the constructed surface passes through a given set of three-dimensional points. On the other hand, approximation techniques, e.g. Bezier surfaces [GOR74b, GIL78, NEW79], use a given set of three-dimensional points as control points affecting the shape of the surface as if they were magnetized points deforming the surface by force of attraction.

The underlying mathematics of the above-mentioned approaches is rather involved, and a significant draw-back of systems based on these surface definition methods is the lack of a truly user-oriented interface shielding users from such tedious details as twist vectors and surface derivatives. An attempt to overcome this difficulty is a recursive surface definition method reported in a recent paper [VEE82].

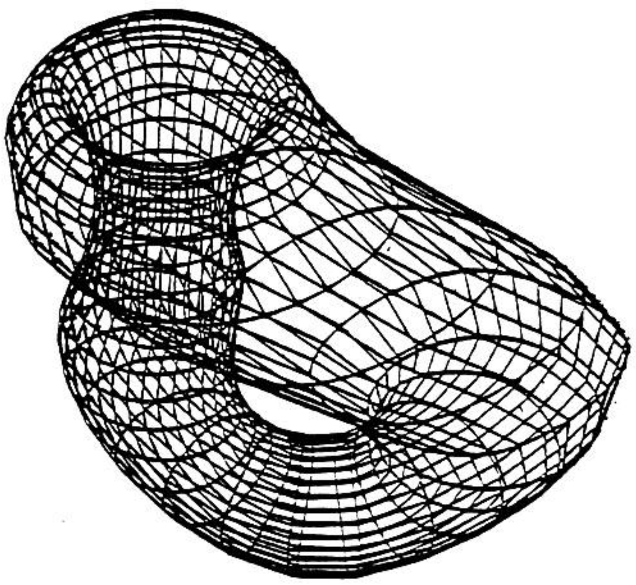


Figure III.13 - Klein Bottle

Validity of a boundary surface representation model as a realizable physical object is a significant issue. Not every set of faces or patches defines a real solid. Figure III.13 shows a well known example of an unrealizable object, an interesting self-intersecting bounded surface with no identifiable inside and outside[2], known as a Klein

2 The picture in Figure III.13 is, of course, a two-dimensional representation of a three-dimensional surface which is an approximation of the four-dimensional object called the Klein bottle. This "object" was in fact defined as a network of

bottle. Several schemes for insuring and checking the validity of boundary models are known [EAS79, BAE78, REQ80b], and are commonly used in practice for the special case of planar-faced polyhedra. In particular, a set of low-level shape operators that define and combine faces, edges, loops and vertices, known as the <u>Euler operators</u> (because they are derived from the proof of the Euler's law) are somewhat popular [EAS79]. These operators are used, for example, in GLIDE [EAS80]. Well-formedness conditions are embedded in the Euler operators, which permit only construction of consistent structures of bounded surfaces that define boundaries of realizable objects. Curved surfaces introduce more complexity into the issue of well-formedness, and generalization of validity schemes that deal with planar-faced polyhedra to cover curved-faced objects is both complex and computationally expensive [REQ80b].

Among different approaches to solid modeling discussed in this section, boundary representation is the closest to a geometric model directly usable for CAM. The reason for this is simple: many manufacturing processes deal with surfaces, e.g. machining operations cut surfaces; edges and vertices emerge as consequences of the meeting of surfaces and cannot be manufactured independently. This means that while edges and vertices, among other things, may be of prime concern during design and analysis, manufacturing cannot deal with these "side-effects" in isolation from surfaces. Many manufacturing processes require detailed information, e.g. calculation of tool paths, which can be extracted rather easily from an explicit surface definition. The

---

Ferguson-Coons surface patches on the NGS system, which shows that this system does not recognize such unrealizable definitions. See Section III.3.3 for NGS.

necessity for identifying surfaces in a final model of design objects for CAM applications forces systems based on other solid modeling schemes to perform a conversion to boundary representation at some point before, for example, they can attempt to generate NC programs.

2.2 Constructive Solid Geometry

The fundamental concept of Constructive Solid Geometry (CSG) is that a (complex) solid object can be represented as "additions" and "subtractions" of simpler solids[4]. Solids are modeled as sets of E^3 points; however, not every set of E^3 points represents a realizable physical object. A definition is therefore needed for sets of E^3 points that represent physical solids. A number of such definitions are possible, but CSG is based on the concept of regular semianalytic sets which, in their general form, are referred to as half-spaces. Informally, a half-space is the set of points on and on one side of a possibly unbounded surface which can be defined by a boolean combination of analytic functions (analyticity of functions is in fact a stronger condition than necessary; see Section IV.1.2). Examples of half-spaces include points on and on one side of a plane, points on and on one side of a (bounded or unbounded) cylinder, etc. Bounded, regular semianalytic sets are called r-sets, and in CSG, it is the r-sets which model real-world solid objects. (Section IV.1.1 explains the necessity for these conditions, their significance, and their implications.

4 Section IV.1 includes a more elaborate discussion of certain formal properties of CSG. On the other hand, the present more general discussion applies to every modeling scheme, including the Realizable Shape Calculus of Section IV.2, that shares the same view of shape composition as CSG.

Limitations of r-sets as formal models for solid objects are discussed in Section IV.1.2.) Examples of r-sets are the set of points on and inside of a bounded cylinder, the set of points on and inside of a bounded polyhedron, etc. "Addition" and "subtraction" of solids are modified boolean set operations union, difference, and intersection which preserve regularity of the sets and, of course, semianalyticity; these operations are called regularized set operations. (See Section IV.1.1 and [REQ77b] for formal definitions.)

There are two CSG schemes, one based on general primitive half-spaces and the other, based on bounded primitive solids. The first scheme permits unbounded half-spaces (such as the one associated with an unbounded plane) and, in addition to regularized union, difference, and intersection, it also permits a regularized version of the set complement operation. This scheme is more general than the second, but since the result of regularized set operations on general half-spaces are not necessarily r-sets, boundedness of the resulting "solids" must be checked separately.

The second scheme deals exclusively with r-sets and since the complement of an r-set is not an r-set, regularized complement is not permitted. In this scheme a solid is modeled as regularized union, intersection, and difference of a given set of predefined r-sets called primitive solids. Each primitive solid itself can be defined in terms of general primitive half-spaces and regularized set operations in the first scheme; but, since r-sets are closed under regularized union, intersection, and difference, unlike in the first scheme, validity of the resulting representations in CSG schemes based on primitive solids

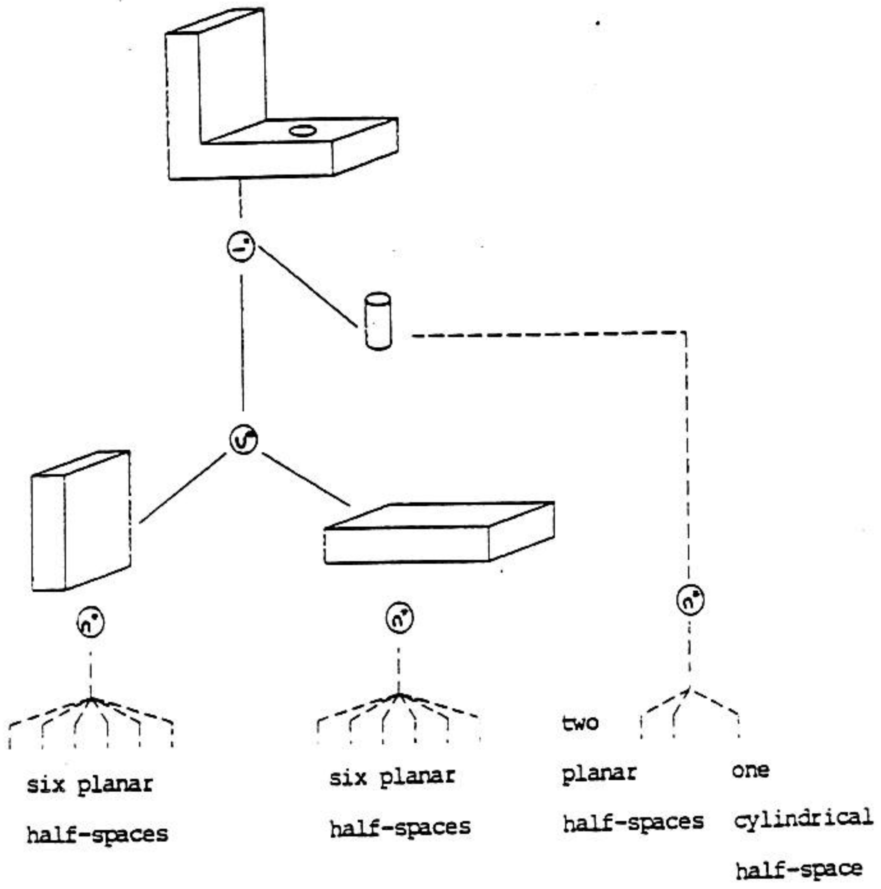


Figure III.14 - CSG Definition of a Bracket

is guaranteed. As seen in Figure III.14, CSG representation of a solid object can be actualized in the form of a tree structure whose leaves are primitive solids (or general half-spaces) and whose branch nodes are regularized set operations.

CSG representations of solid objects are complete and can be made unambiguous (see Section IV.1), but they are not unique (there are infinitely many other possible representations for the bracket of Figure III.15, for example). CSG schemes have shown promise as suitable models for solid objects and a number of powerful experimental and commercial CAD systems based on CSG are in current use, including PADL [PAD74], GMSOLID [BOY80, BOY82], GDP/GRIN [WES80a, FIT81a], and Synthavision [GOL79] which use primitive solids and TIPS [OKI73, OKI76] which uses half-spaces.

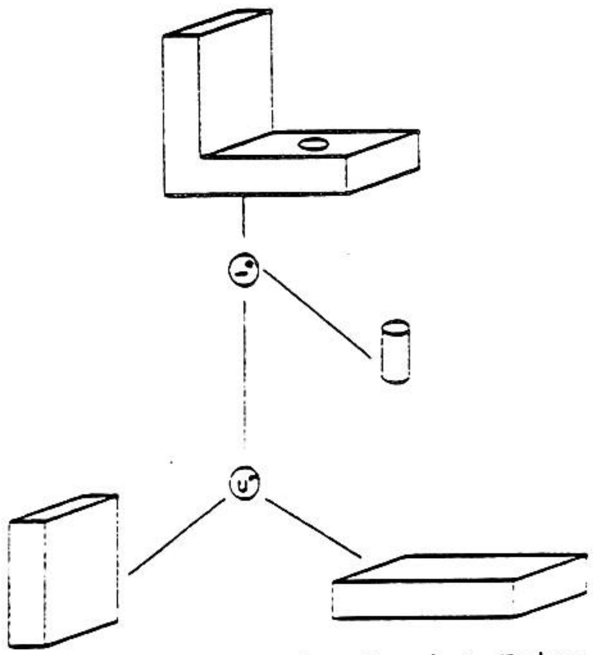


Figure III.15 - CSG Definition of a Bracket Using Primitive Solids

Primitive solids based CSG schemes tend to yield the most compact representations for simple solid objects in comparison to, for example, boundary representation or half-space based CSG schemes. However, for complex solids the number of leaf nodes in a primitive solids based CSG tree tends to be equal to the number of their distinct surfaces and not only this advantage of compactness of representation fades away, but an additional burden of irrelevant information also creeps in. The reason behind this is that as objects' shapes get more and more complex, the number of primitive solids in their definitions increases, while the shapes of these constituent primitives tend to bear less and less resemblance to the final shapes of objects. Instead, the contribution of primitives to the final shape becomes more and more local, approaching the extreme of one primitive solid contributing one face per distinct surface. On the other hand, the amount of information

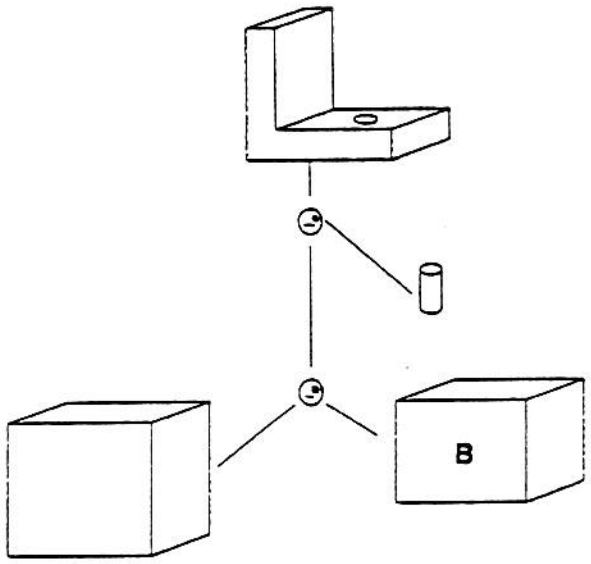


Figure III.16 - Manufacturing-Oriented CSG Definition of a Bracket

contained in the definition of a (primitive) solid is generally more than that of the definition of any of its faces, and this is true regardless of the fact that representation or definition of a (primitive) solid may be made even more compact than that of its bounding surfaces. This means that by using a primitive solid to define a surface (instead of defining the surface more directly) one includes into the model some irrelevant information which makes the model more complicated than necessary. For example, consider the bracket of Figure III.15 again. Many people would actually define this bracket as depicted in Figure III.16, rather than Figure III.15, which happens to be a more "natural" definition since it is much closer to the actual way in which the bracket is built. Note that the length of the cylinder in Figures III.15 and III.16 and length, height, and width of block B in Figure III.16 are in fact irrelevant information; they must be greater than a required minimum, but any definite number greater or equal to this minimum is "as good as" (and as irrelevant as) any other.

Furthermore, the "shape information" defining four out of six faces of block B and their orientation angles, is also irrelevant.

The presence of this type of irrelevant data in CSG models not only causes mental obstacles for designers who must nevertheless generate them, but more seriously, produces a complex problem for application programs using such models which must, quite non-trivially, "discover" and discard this information. In practice, the expense of discarding irrelevant information is paid at the time of conversion of CSG trees to boundary representation.

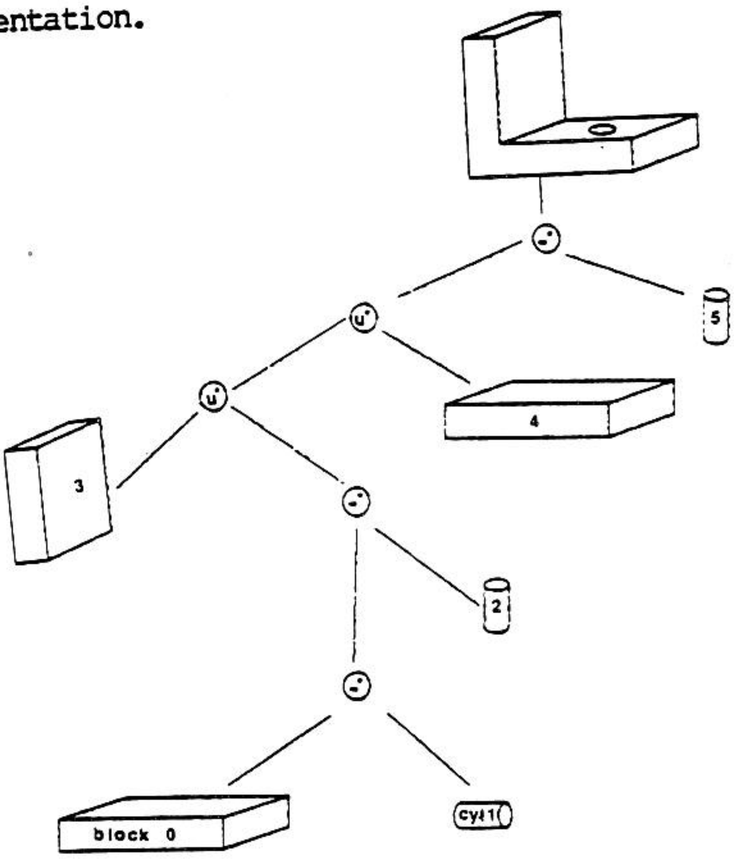


Figure III.17 - An Unnecessarily Complex CSG Definition

Another draw-back of approaches based on primitive solids is that arbitrary sculptured surfaces become awkward or impossible to represent. In addition, definition of solid objects in terms of primitive solids is somewhat "unnatural" for many people compared to more traditional methods. At the same time, CSG representation of a part may bear no relationship to the operations required for its fabrication either. In particular, the union operation causes the most difficulty because in its general form, it has no corresponding manufacturing process. As a first example of the type of problems that arise, consider the bracket of Figure III.15 again. Strictly speaking, the CSG definition of this bracket in Figure III.15 is "as good as" any other such simple CSG definition, such as the one in Figure III.16. In an integrated CAD/CAM system, however, the definition of Figure III.16 would be the preferred definition because it resembles the actual machining process of fabrication of the bracket, and because it is much more suitable for extraction of information such as specification of the required raw stock and the manufacturing process plan. With no criteria built into the modeling scheme (CSG), however, neither definition can be "analytically" distinguished as the better or the best among many equally acceptable definitions for a part.

A more serious problem associated with the union operation in CSG is illustrated by the CSG tree in Figure III.17, which with an appropriate set of metric data defines, in a "messy" style, the same bracket as in Figure III.15. In this figure, a designer starts by subtracting cylinder 1 from block 0; this is an error and is later corrected by the designer who now proceeds to make the correct hole by subtracting cylinder 2. Next, block 3 is added to form the L-shape and then the designer attempts to correct his initial mistake by "filling-up" the first hole using block 4. addition of this block, however, also partially fills up the correct hole which must be "drilled" again. Subtraction of cylinder 5 accomplishes this, but it also makes the earlier subtraction of cylinder 2 redundant. The problem with the resulting tree is that it is by now a counter-intuitive and "dirty" representation full of redundancy. To underline the issue here, consider the problem of defining a process plan based on the tree in Figure III.17.

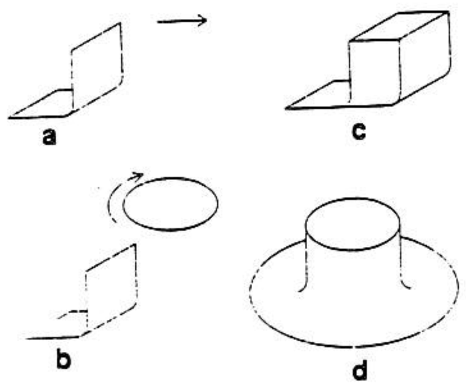
A reason why an operator like union is desirable is that it somewhat resembles assembly operations. But regularized union is far more powerful and general than an "assembly operator", and unless restricted, it can lead to definition trees orders of magnitude messier than our simple example above, and many times more complicated than corresponding "proper" CSG trees defining the same parts. (At the same time, regularized union fails to consistently satisfy our intuitive expectations of an assembly operator. See Sections IV.1.3 and IV.1.4.)

An alternative to imposition of restrictions on application of union is to develop methods of "simplifying" CSG trees to equivalent trees where volumetric features are represented in a "standardized style" or a "normal form". Some results are available on feature extraction from and decomposition of volumetric representations [WOO76, WOO82]. A recent algorithm, for example, decomposes a boundary representation into an alternating sign series of volumetric entities, i.e. an alternating addition/subtraction of solid components [WOO82]. Solid components are computed by the algorithm as the convex hull of a partial result which diminishes as the series converges to the original object. It is not clear, however, whether these approaches can be generalized to deal with additional constraints on, for example, shape and number of solid components; criteria of interest in simplification of a CSG tree or its conversion to a normal form.

Aside from its shortcomings (see Section IV.1), CSG is a formalism for modeling of solid objects, irrespective of the way in which they can be manufactured or functionality of their components. A CSG definition, in general, should therefore be seen as a "declarative" representation of an object's shape in the sense that the set operators (union in particular) in a CSG tree cannot be interpreted as operations producing solids out of solids, but rather, as statements of factual relations between somewhat arbitrary shape components which do not necessarily have any functional or manufacturing-related significance or association with one another. As such, CSG can be dubbed a "calculus of shapes" that may strengthen the backbone of solid modeling systems, but, would better be kept away from end-users, not just because of issues like naturalness, friendliness, or orientation of its operators, but also because its power needs to be tamed with structure and discipline. This can be achieved by reintroducing some type of a "context" which is completely abandoned by CSG through its high level of abstraction.

2.3 Sweeping

The principle of sweeping is intuitively very simple: a set of points called the moving object or the sweeping set, moving through space along a curve called the trajectory, sweeps a "volume"; thus a sweeping set and a trajectory curve can represent a solid object.



(a) and (b): Two-dimensional surfaces and sweeping path curves

(c) and (d): Corresponding three-dimensional entities resulting from sweeping

Figure III.18 - Invalid Objects Created by Sweeping

The sweeping set could be rigid, i.e. maintain the metric properties of its members as it moves along the trajectory curve, or non-rigid, i.e. get deformed as it moves. Non-rigid sweeping is used, for example, in modeling of curved objects in a system for three-dimensional image analysis where objects like toy dolls and animals are decomposed into a set of tube-like shapes, called generalized cylinders, each of which is defined by a space curve and a circular cross section function on this axis curve [AGI76].

A three-dimensional solid can be defined by sweeping a two- or a three-dimensional solid set along a trajectory. Sweeping a three-dimensional solid always produces a valid three-dimensional solid. This type of sweeping is very important in modeling of material removal processes, such as machining, where a solid cutter sweeps along a cutting path removing the swept volume. Sweeping of solids is also important for dynamic interference checking.

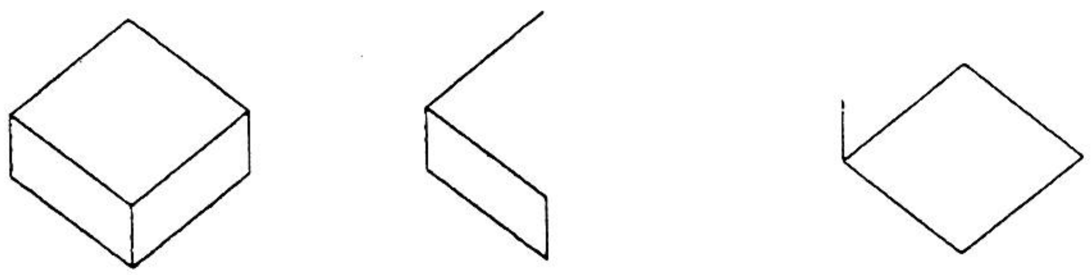


Figure III.19 - Multiple Sweeping Definitions for an Object

The conditions under which the result of the sweeping of a homogeneous two-dimensional solid (a bounded surface) is guaranteed to be a homogeneous three-dimensional solid (i.e. a valid object) are not known [REQ80b]. For example, if the trajectory curve is "parallel" to (a segment of) the sweeping set, the resulting solid will include "dangling" faces, as shown in Figure III.18.

When the sweeping set is a planar rigid solid polygon (possibly, with curved edges and holes) and the trajectory is a line segment not parallel to the polygon's plane, the scheme is called translational sweeping. Likewise, when the trajectory is a circular arc, the scheme is called rotational sweeping. Validity of the results of translational and rotational sweeping as three-dimensional solids can be ascertained by checking the validity of the sweeping sets as planar solid polygons.

One of the problems with sweeping is that representations of solids are not unique. This is true even in the restricted case of translational sweeping, as seen in Figure III.19. Another draw-back of this technique is that sweeping representations are not quite suitable for direct manipulation by many important algorithms, including for example, shape modification operations.

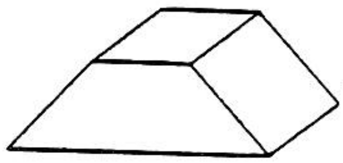


Figure III.20 - An Object with no Translational Sweeping Definition

Translational and rotational sweeping are sometimes very convenient to use. The domain of translational and rotational sweeping is, however, limited and not every solid can be represented by these schemes, e.g. the object in Figure III.20 cannot be defined by translational or rotational sweeping alone. Rotational sweeping is particularly important in modeling of turned parts. Translational sweeping is important in modeling of sheet metal parts. Rotational sweeping definition of turned parts happens to be compatible with the actual lathing process of their fabrication. This makes it possible to use the rotational sweeping model of these parts for both CAD and CAM applications. For example, different contours of a rotational part can be extracted from such a model directly. Analysis of these contours is important in determination of the number, type, and the cutter path of the turnings required for machining of the piece. Certain sheet metal manufacturing processes can also take advantage of a translational sweeping definition of sheet metal parts.

Translational and rotational sweeping schemes are sometimes used in conjunction with other solid modeling schemes to provide both the convenience of sweeping when it seems natural, and the richer domains not covered by pure sweeping alone. For example, GDP/GRIN and GMSOLID, two CSG based systems, permit users to define their own primitive solids through sweeping [FIT81a, BOY80]. These solids are then treated like any other primitive solid and can be combined with other solids using regular CSG-type operations. Adoption of sweeping in such systems, nevertheless, is essentially as a convenient input mechanism rather than an alternate representation model, in that solids defined through sweeping are "evaluated" on input and kept in a different format, usually, boundary representation.

3 Typical CAD/CAM Systems

In this section we consider a number of typical CAD/CAM systems aimed specifically at design and manufacturing of mechanical parts. Our primary purpose is to gain a perspective of the evolution of CAD/CAM systems and to see the direction of this evolution.

First, we will consider CADAM because it is one of the earliest systems developed and has been in use for over a decade, and because it is typical of an entire family of systems based on traditional drafting methods. Straightforward extension of this type of systems to three dimensions gave rise to another family of CAD/CAM systems that primarily use wire-frames. The need for three-dimensional sculptured surfaces specifically in the aerospace industry, produced a different family of systems that deal with true three-dimensional surfaces, typical of which is the NGS system discussed below. The necessity of "solid modeling" (i.e. representing objects and their contained volumes) brought about systems like PADL, GDP/GRIN, and GMSOLID which are discussed below. PADL is the forerunner of these systems and typifies a Constructive Solid Geometry approach to definition of mechanical objects, while GDP/GRIN and GMSOLID each have interesting aspects in their user interface and internal representation methods.

3.1 CADAM

CADAM was originally designed and developed by the Lockheed-California Company to facilitate the production of engineering drawings using classical two-dimensional projective geometry [CAD80]. It is a software package over 80% of which is written in Fortran and the rest in

Assembly, and runs on IBM 370-type main-frame using, primarily, IBM 3250-type vector refresh graphic terminals. Views as orthographic projections of an object are a central concept in CADAM; special facilities are included to assist in construction of views, including oblique and isometric, once a first view is created. Although until recently, CADAM was a strictly two-dimensional system, prime attention has been paid to design of mechanical parts. A finite element modeling package and three-dimensional splines and ruled and bicubic surfaces have been added rather recently, but up to its current release, CADAM still remains a two-dimensional system in spirit. A full three-dimensional version of CADAM is soon to be released which will maintain upward compatibility with previous two-dimensional releases, while providing the same set of functional capabilities as wire-frame systems.

Internally, CADAM is an interactive interrupt-driven package intented for a middle to large main-frame. Users' interactions with CADAM are through an alphanumeric keyboard, a set of function keys, and a vector refresh terminal on which menu items and graphic entities can be selected by a light-pen. CADAM's data consists of elements like points, lines, arcs, ellipses, splines, dimensions, notes and text, surfaces and three-dimensional curves, and points, details, and view transformation matrices. Formats in which information is kept is a compromise between modifiability and efficiency of graphic representation. CADAM's "data base", which contains all such information pertinent to a single drawing is not a true data base, but rather, it is a "flat file", i.e., a sequential collection of data that does not represent inter-element relationships.

Since CADAM was developed in the late 1960's and has evolved for over a decade in real production environments, its user interface is quite rich and pleasing in function. However, the interface itself, i.e. the syntax of user-system communication, leaves much to be desired with its inconsistent behavior from function to function, inconsistent prompts, messages, and menus, and sometimes, not-very-logical grouping of functions under function keys and in menus. As compared to other systems, CADAM makes very good use of menus and light-pen, to the extent that typing is only occasionally required. The light-pen performs two different tasks in CADAM: picking an existing entity visible and accessible on the screen, and pointing to an approximate location or a direction; in CADAM jargon these operations are called selecting and indicating, respectively.

A CADAM drawing is a 20,000 x 20,000 units virtual piece of paper which a user sees through a "window". Function key WINDOW permits users to position their windows (i.e. terminal screens) such that they see desired parts of their drawings in the size and orientation they choose. Operations performed under WINDOW in no way affect a drawing itself, but rather, they modify a user's view of his drawing, i.e. what he sees through his window.

Drawings are independent entities which are stored individually into users' private libraries. A drawing itself consists of a number of views, which presumably represent planar projections of an object each with its own independent coordinate system. Views, of course, may be invalid "projections", since they may not actually correspond to one another or to a real object and can contain three-dimensional entities.

Views consist of elements and, while their relative scale, orientation, and position on a drawing are independent of one another and can be changed whenever desired without affecting other views, CADAM retains views' logical relationships based on the lines of sight and projection planes used to create them.

Twenty six out of the thirty two keys available on the function keyboard are used in CADAM. With the exception of two function keys ("Yes/No" and INDICATE) whose effects are "instantaneous", pressing a function key changes CADAM's mode of operation. Each mode is named after its corresponding function key, and, under each function (mode) there is a menu of sub-functions available, and sometimes, certain menu items have sub-menus of their own.

CADAM operation consists of cycles of pressing a function key, selecting one or two menu items, and then supplying required parameters or operands to the operation thus specified. Parameters and operands are given by using the light-pen, by pressing a function key called "Yes/No" to accept a default value, and rarely, by typing in a value, a name or some text.

Function keys POINT, LINE, CIRCLE, and SPLINE make it possible to define basic geometric entities in a wide variety of ways. Function key TYPE changes the line-type (sometimes called "font" in other graphic systems) of an element to dashed, phantom, solid, or N/C, and its weight to high, medium, or low. RELIMIT and CORNER modify existing geometric entities; e.g. an element can be broken into two pieces, each becoming a separate entity, under RELIMIT, which also provides the ability to trim or extend an element up to another intersecting entity. Generally, any two elements that (potentially) intersect can have their intersection finished with a convex, concave, pigtail, or sharp corner under function key CORNER, with the parts of elements extending beyond this intersection trimmed, if so desired. CORNER also includes menus for easy construction of machine and sheet-metal jog shapes.

Function key OFFSET duplicates elements at given offset distances. Aggregates of elements may be formed into a group under the function key GROUP and may subsequently be referenced collectively under a number of other function keys. GROUP also makes it possible to translate, rotate, scale, and mirror elements or groups. DETAIL and SYMBOL create and use aggregates of elements which are referenced by an index and are stored in a common symbol library or are local to a drawing. DIMENSION places dimensions on drawings; CADAM, naturally, "knows" the geometric properties of elements, and a user must only tell CADAM what element is to be dimensioned and where is the dimension to be placed.

Function keys MISC and MISC-2 provide features like creation and editing of notes and text, cross hatching and filling, arrows, deltas, balloons, etc., while the N/C function generates numerical control programs directly from geometric information available in a drawing. SURFACE creates three-dimensional splines and ruled and bicubic surface patches, although its capabilities are limited and somewhat tedious to use. Function keys ANAL and ANAL-2 provide facilities for analysis of individual elements, computation of volume, weight, and density, calculation of area and other sectional properties of a closed contour, and limited capabilities for beam-load, torsion, lug shear and bearing, crippling, and fluid delivery analyses.

Function key FILES performs housekeeping functions such as logging in and out, starting a new drawing, saving, recalling, and plotting a drawing, etc. SHOW "hides" or erases elements. AUX-VIEW creates new views and selects a particular view as the current view, and includes menu items for projection of elements from one or simultaneously two other views onto the current view. With these features, one can essentially create any orthographic or oblique projection views, including, in particular, isometrics. However, although simple tasks such as construction of orthometric views are relatively simple and convenient to perform, the process of creation of more complex views such as isometrics is conceptually difficult and requires rather tedious user involvement.

Function key ORIGIN creates auxiliary coordinate systems within a view, but unfortunately, such auxiliary axis blocks are not consistently usable under some other function keys. MESH provides facilities for creation of two- and three-dimensional finite element models and also includes a not yet released three-dimensional piping package.

At its lowest level, CADAM uses standard IBM software for vector refresh graphics (GAM) and is thus somewhat device independent. However, the system architecture is capable of presenting only two facets to its users: an external appearance as a package of hardware and software, and an internal representation of the data in its data base. In other words, users of CADAM can either utilize it as a complete system or, if they require some modification (e.g. in order to interface

CADAM to other application packages or to include certain specialized functions), they suddenly must deal with the internal details of the information representations in CADAM. Sophisticated modifications that might involve CADAM's interrupt protocol and its overlay structure, such as changes in the use of the light-pen, function keys, or the alphanumeric keyboard, are formidable for a user to consider because of their drastic global interrelationship with the overall system structure.

Because CADAM's architecture is not layered into functionally independent components with formally defined interfaces, interaction among program units of CADAM is generally unstructured, and logical functions are dispersed. Such an architecture provides no intermediate points of entry and cannot support communication at any level below its surface. Therefore, the only "easy" way for exchange of information between CADAM and the external world, other than its top level, is through its data base. Again, since data base related functions of CADAM do not form a logical unit and inter-element relationships are not recorded, the data base is a flat file with no safeguards against violation of integrity and semantics of represented models.

A somewhat recent enhancement to CADAM is the Interactive User Exit (IUE) which permits on-line invocation of user-written application packages during an interactive CADAM session [IUE82]. IUE or any similar enhancement cannot provide a new level of communication because, in principle, the problem is an architectural barrier that can be crossed only by a major structural redesign; IUE simply supports interactive user-application communication with CADAM at the same two possible levels.

Although CADAM supports some three-dimensional design, its orientation is toward two-dimensional computerized drafting where the end result is a set of orthographic projections comprising engineering drawings to be used for manufacturing.

3.2 Wire-Frame Systems

Although wire-frames are very popular feature of commercial CAD/CAM systems, there are no commonly used pure wire-frame systems, due to inherent problems and limitations of this model. Wire-frame based systems augmented by certain somewhat limited surface geometry capabilities are common, however, with or without extensive drafting-oriented functions extended to three-dimensions. Most commercially available three-dimensional turnkey CAD/CAM systems fall into this category, as they lack coherent automatic capabilities for insuring integrity of solid objects, and, usually, are not very sophisticated in their handling of free-form surfaces.

Extension of two-dimensional drafting functions to three dimensions can be done in a number of ways. One is to work directly with three coordinate values, instead of two. This approach, however, is not always convenient, especially for people who are used to two-dimensional thinking. Many systems, consequently, provide another mode of operation where, essentially, one coordinate can be "factored out" dynamically from the user-system communication. This is done, simply, by introducing the concept of a "current work plane" which permits users to carry on a two-dimensional dialog with the system relative to a local coordinate system on the work plane, while a transparent conversion to the master three-dimensional coordinate system is performed automatically.

For example, a new three-dimensional release of CADAM extends its two-dimensional functionality to three dimensions by permitting users to select a design plane as "current". The current plane may be changed at will, and every defined entity is stored as a three-dimensional geometric element. All of the functionality of two-dimensional CADAM is available as it applies to the current plane. In addition, many CADAM functions are extended to deal with off-plane three-dimensional entities as well. For example, an additional feature under function key OFFSET permits off-plane offset of geometry. Basic geometric entities such as points, circles, etc., can be created in three dimensions directly, and intersections of elements can be found in the out-of-plane mode.

It is beyond the scope of this section to detail operations of specific systems, since they are close enough to that of CADAM in philosophy. Functionally, most three-dimensional software package and turnkey systems additionally provide a repertoire of drafting operations similar in concept to those available in two-dimensional CADAM, and a variety of surface construction facilities in the same spirit as those available in surface oriented systems such as NGS (which is described in Section III.3.3). Three-dimensional viewing functions are sometimes included in work-station hardware and projection capabilities for construction of orthometric, isometric, oblique, and various types of perspective views are also often supplied. A pleasant feature of some of these systems, e.g. Calma [DDM80] and Computervision [DVS82], is the ability to tie projection views to the three-dimensional model dynamically such that modifications to the model or to an appropriate number of views automatically are reflected both in the model and in all other projection views.

Another departure from CADAM-type of functionality in many systems is in their user interface. Many systems provide some type of macro definition capability at their command level, and, usually, those utilizing a digitizing tablet also permit dynamic assignment of commands and macros to flexible user-definable menus on the tablet. Some other systems, e.g. Applicon [AGS78], also permit assignment of macros and commands to function keyboards similar to CADAM's. An additional level of user-system communication is provided in systems such as McAuto's Unigraphics [GRA80], where the power of the system as a geometric and graphic package is made available through a programming language environment as well. In Unigraphics, this language is called GRIP [GRI80]. GRIP is a Fortran-like language which permits creation and manipulation of geometric and non-geometric entities, file management operations, arithmetic operations, interrogation of the Unigraphics data base, display of messages and menus, and interaction with users. GRIP programs can be compiled and integrated into a running Unigraphics system. GRIP, thus, can be used to customize a system's interface through modifications ranging from simple enhancements to standard system functions all the way to inclusion of elaborate primitive instancing family of parts programs that automatically generate complete geometric models and engineering drawings while leading interactive users for input of critical family-specific parameters. Such tasks, however, require intensive programming effort and familiarity with internal aspects of the system and are thus beyond the scope of most non-programmer end-users.

3.3 NGS

NGS, for "a Numerical Geometry System" [NGS80, BUR75, MAY76], was jointly developed by IBM and Teledyne-Ryan Aeronautical in the mid 1970's primarily for lofting applications in the aerospace industry. Although IBM recently dropped support of NGS in the United States, it is currently used in Europe. NGS is a software package written almost entirely in PL/I and runs on an IBM 370-type main-frame using IBM 3277 alphanumeric terminals with Tektronix storage tube graphic attachments for its user interaction and graphic output. It provides non-programmer users with comprehensive design capabilities for complex three-dimensional surfaces that have historically required difficult and time-consuming manual operations. Users interact with the system in an engineering-oriented definition and command language, and, once objects have been defined, calculations of interest to engineering analysis, design, test, and manufacturing groups using NGS geometric operators can be performed. Modifications to defined surfaces can be made quickly by modifying their control points, and NC programs for mill cutting of dies used for press shaping of flat metal sheets can be generated automatically.

Geometric definition capabilities include three-dimensional points, lines, splines, planes, and Bezier and Ferguson-Coons surface patches. For the sake of internal uniformity and simplicity of algorithms, all entities are represented in terms of spline polynomials and surface patches. In particular, conic sections and conic surfaces are not available analytically and are approximated by splines and Ferguson-Coons patches, respectively. Every geometric entity (or collection of entities) must be given a name upon definition. These entities are stored separately in a "data base" (called a work-area) and the system, in general, is not aware of their inter-relationships. Automatic filleting and duct definition (with Ferguson-Coons patches), calculation of patch-patch intersection, numerical control tape generation, volume, cross-sectional area, and many other analysis functions are included. Data base interaction allows entities to be accessed, modified, saved, and protected. Display functions include three-dimensional rotation, translation, windowing, hidden-line removal (with sculptured surfaces), and several others.

Each user's defined geometric entities reside in a private work-area which remains intact between sessions. This applies only to present "stand-alone" implementations; reorganizations to adapt NGS to generalized data base management systems require no conceptual level modifications, and such a system's inherent integrity and consistency checks would then apply to NGS-defined geometry. NGS commands can be classified into four categories: geometric entity creation commands, entity management commands, commands for computation of geometric properties, and other miscellaneous commands. Commands in the first category permit creation of points, lines, piecewise-polynomial curves, planes, bounded planar regions, Ferguson-Coons patches, Bernstein-Bezier patches, fillets, and axially-symmetric surfaces by revolution of piecewise-polynomial curves. Furthermore, geometric entities can be created by intersecting other entities and by mirroring, moving, and rotating existing ones. In addition, special effects can be created through user-supplied three-dimensional transformation matrices.

Entity management commands are similar to directory management operations in most interactive systems. They permit listing, deleting, and renaming of entities, make it possible to group and store entities into files, and to retrieve them back. In addition, several geometric entities can collectively be given a name thus forming a new super-entity. The third category of commands permits computation of any curve length, any finite surface area, any bounded region's volume, volume and surface centroids, minimum distances between entities, and moments of inertia. Other miscellaneous commands allow various display (e.g. scaling, three-dimensional windowing, etc.) and plot functions including removal of hidden lines, editing of descriptive texts associated with entities, and NC programming.

The user interface of NGS is poor; each command must laboriously be typed in, the command syntax is very simple and inflexible, there are no menus, macros, or function keys, and there are no means for "graphic input" to the system. This makes the system rather cumbersome to use and disguises the fact that, underneath, NGS is a powerful surface oriented system.

Geometric "solids" are defined and stored as a collection of surfaces [NGS80]. For example, an engineer could interpolate a Ferguson-Coons surface or fit a Bernstein-Bezier patch onto a 4 by 3 rectangular grid of three-dimensional points by entering

FCNET or BBP LOFTSURF (4 3) point_1, point_2, ..., point_{12}

(where each point_i is entered as its corresponding X, Y, Z coordinates) and an analytic representation of the surface would be saved as LOFTSURF in the engineer's work-area. Both of these surface types are defined by and stored as boundary information (and implicit associativity relationships between patches in the case of a Ferguson surface). LOFTSURF could subsequently be modified and re-fit, intersected with other three-dimensional entities, analyzed, or plotted. There are, however, no provisions in the system for checking the well-formedness of collections of surfaces and unrealizable objects like the Klein bottle of Figure III.13 can be generated in NGS (see Section III.2.1). Thus, by the taxonometric classification proposed in [HIL72b], NGS is a member of the family of surface oriented systems and does not qualify as a solid modeler based on boundary representation.

Assumptions about end-users and the overall approach in NGS are appropriate for sculptured surface definition and manipulation and, hence, for lofting. CADAM, on the other hand, provides much more powerful mechanical drafting capabilities for production of manufacturing drawings. While the underlying mathematics of NGS is necessarily complex, its user interface is much more primitive than CADAM's, and certain drafting functions, such as dimensioning, are not available in NGS at all. There is a software package available that allows transfer of drawings between CADAM and NGS.

3.4 CSG-Based Systems

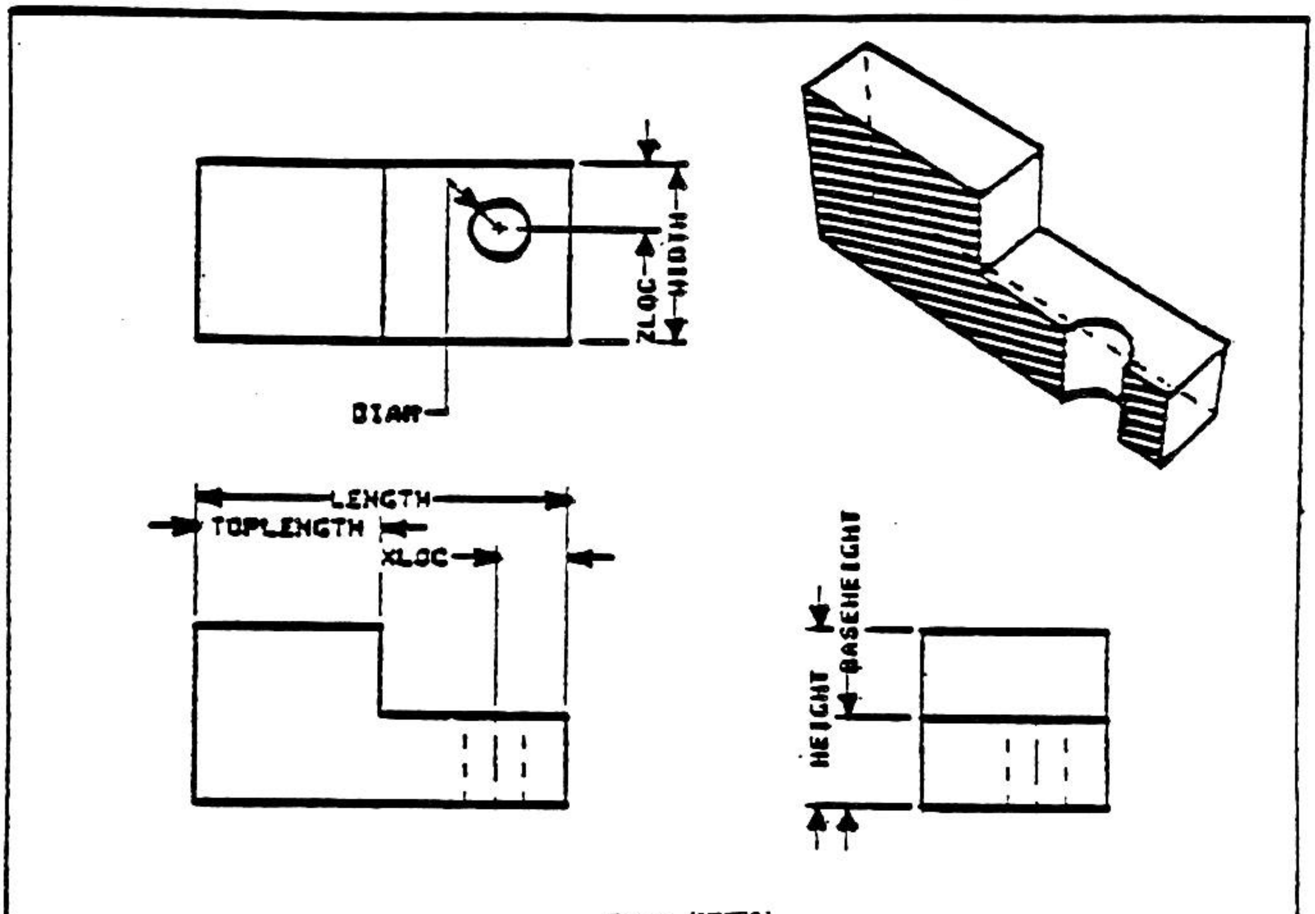
Rather recently, a number of "solid modeling systems" have evolved, both as research tools and as production CAD systems, which at least partially, are based on a CSG modeling scheme. PADL, GMSOLID, and GDP/GRIN (which apparently, is about to be officially renamed IBMSolid), are three of the CSG-based systems reviewed below. Other such systems include TIPS [OKI73, OKI76], Synthavision [GOL79], GLIDE [EAS80], BUILD-2, DESIGN, and ROMULUS [HIL82a], for some of which CSG is <u>not</u> the only and/or primary internal solid representation model.

3.4.1 PADL

PADL, for Part and Assembly Description Language, is a language designed by the Production Automation Project, University of Rochester, for defining solid objects using a primitive solids based CSG model [PAD74, VOE78]. The first processor for this language, PADL-1, was implemented from 1975 through 1977, mainly as an experimental system to demonstrate feasibility of new ideas and algorithms, and to serve as a tool for education. PADL-1 consists of some 70,000 lines of Fortran, about half of which are comments, and accepts PADL statements either in batch or from a keyboard interactively, producing line drawings of objects on graphic terminals or plotting devices.

PADL has two types of statements: definitional statements and commands [VOE78]. Definitional statements define a name and associate an object or a value to that name in a Fortran-like assignment statement syntax. Definitional statements are <u>not</u>, however, assignment statements in the same sense as in regular programming languages, and PADL names are not variable names, since once a name is defined and associated with a value or an object, redefinition or reassociation of that name is not permitted. The order of definitional statements is, therefore, irrelevant, i.e. a name may be used before it is defined and associated with a value. The right-hand-side of a definitional statement is, in general, an expression. There are two types of expressions, object expressions and value expressions. Value expressions deal essentially with numbers (and special numeric sub-types called distances, tolerances, and distance-chains), and the usual arithmetic operations are provided, together with three special operators for manipulation of numeric sub-types. Object expressions are those resulting in an object, i.e. a solid. Object manipulation operators consist of those for CSG operations of union, intersection, and difference, an assembly operator which collects objects under a single name, and a move operator for translational positioning of objects.

Commands can appear anywhere in a PADL program and are executed immediately, if possible. There are three types of commands in PADL-1: utility commands such as those for saving and recalling PADL definitions into and from files, editing commands which permit a BASIC style editing of PADL source statement definitions, and finally, graphic commands that permit display (on a CRT) and drawing (on a plotter) of a graphic representation of objects. Various parameters and qualifiers to these graphic commands permit creation of different orthometric projection views and isometrics with desired scales, automatic posting of dimensions, treatment of hidden lines, etc.



From: [VOE78]

Figure III.21 - Projections of a Part Produced by PADL

PADL-1 offers two primitive solids, blocks and cylinders. The edges of blocks and the axes of cylinders must be parallel to the axes of the master coordinate system in PADL-1, i.e. all primitives must be orthogonal. Figure III.21 shows the three standard orthometric views of an L-shaped bracket, together with a sectioned isometric, produced on PADL-1; PADL statements and commands that define and display these views are shown in Figure III.22.

PADL-1 is not a suitable system for real production environments, and it was not meant to be. In addition to the limitations caused by the fact that only two primitives are supported and that they can only be combined orthogonally (no rotation), the syntax is inflexible and

```
002    :
004    : FIRST DEFINE TWO BLOCKS TO FORM THE BASE AND TOP
006    : AND THEN UNION THEM. NOTE THAT THE TWO BLOCKS
007    : INTERPENETRATE. (THE ORIGIN IS THE DEFAULT LOCATION.)
008    :
010         8BASE = $B(LENGTH, BASEHEIGHT, WIDTH)
020         LENGTH = 4
030         BASEHEIGHT = 1
040         WIDTH = 2
045    :
050         8TOP = $B(TOPLENGTH, HEIGHT, WIDTH)
060         TOPLENGTH = 2
070         HEIGHT = 2
073    :
100         8LSHAPE = 8BASE .UN. 8TOP
102    :
104    : NOW DEFINE THE HOLE AS A SOLID, LOCATE IT PROPERLY,
106    : AND DIFFERENCE IT FROM 8LSHAPE. (NIL IS A NULL DISTANCE.)
108    :
110         8HOLE = $CY(DIAM, BASEHEIGHT)
                 AT (LENGTH -' XLOC, NIL, ZLOC)
120         DIAM = 0.625 : PM(.005)
130         XLOC = 0.75 : PM(.001)
140         ZLOC = XLOC
142    :
150         8PART = 8LSHAPE .DIF  8HOLE
152    :
154    : ASSIGN A DEFAULT TOLERANCE TO ALL DISTANCES NOT
156    : EXPLICITLY TOLERANCED ABOVE
158    :
160         $DEFTOL = PM(.01)
162    :
164    : NOW PRODUCE FIG. 2 VIA A DISPLAY COMMAND
168    :
170         DISP(8PART), O3R, DLABEL: AUX(SECT(8PART,$Z,0.75)
```

From: [VOE78]

Figure III.22 - PADL Definition of a Bracket

awkward to use. A user-friendly interface replacing its harsh syntax with a "graphic syntax" of picking, pointing, and menus, allowing flexible manipulation and editing of entities without exposing users to PADL's textual definitions is a necessity[5]. Addition of more primitives, e.g. cones and spheres, to PADL-1 is conceptually simple and would increase geometric coverage of the system. Removal of the orthogonality restriction and inclusion of a rotation operator, are both conceptually straightforward and necessary. The latter, however, requires conceptual modifications to the so-called dimensional trees of

5 An attempt to create such an interface to PADL is a presently on-going project at McAuto, Cypress, California.

PADL-1, which are the basic structures for representation of dimensions and tolerances [REQ77a]. Handling of dimensions and tolerances in the general case requires the ability to access and "name" the bounding faces of objects. This, in particular, is in conceptual conflict with the premise of primitive solid building blocks, and raises yet another question as to the appropriateness of primitive solids based CSG as a model for end-users.

Another PADL processor, PADL-2, was developed between 1979 and 1981 by the Production Automation Project. On the opposite philosophical extreme of turnkey systems, PADL-2 is an evolving collection of software modules written in Fortran, aimed at providing a common core of representational and computational facilities for a variety of solid modeling systems suitable for different application environments.

An example of an integrated system utilizing PADL-2 modules is a demonstration software system, called P2/MM, which adds various input translation/editing and graphic output facilities to the PADL-2 package. An interesting input interface to P2/MM, in particular, is an extension of a package from Leeds University which accepts rough two-dimensional outlines drawn by a curser on a CRT, fits the outline with straight lines and circular arcs, and automatically produces PADL-2 definitions for solids resulting from translational sweeping of such closed contours in terms of combinations of blocks, wedges, and cylinders [BRO82]. The main user-interface in P2/MM, however, is a PADL-1 style textual language with slightly improved syntax and some additional capabilities, including a subprogram-like generic object definition facility.

Geometric coverage of PADL-2 is broader than PADL-1 and, with its richer set of primitive solids, can represent any solid bounded by any combination of arbitrarily-oriented planar, cylindrical, spherical, and conical surfaces. A variational subsystem for handling of dimensions and tolerances is planned for PADL-2, but it is not to be implemented in the near future [BRO82].

3.4.2 GMSOLID

GMSOLID is a solid modeling system which has benefited from the PADL experiment. In 1966, General Motors began development of a system called CADANCE (for Computer Aided Design And Numerical Control Effort) which came into production use in 1971. CADANCE is oriented towards three-dimensional curves and surfaces and is used primarily in sheet metal body-panel design at GM [MYE82]. The underlying approach of CADANCE is, however, not suitable for certain applications which require volumetric entities. An example at GM is the problem of packing luggage into the trunk of a newly designed car. This brought the attention of GM Research Laboratories to solid modeling and lead to development of GMSOLID in the late 1970's, based on some of the results of the PADL project [BOY79a, BOY80, BOY82].

GMSOLID is a CSG-based production-quality solid modeling system with powerful design and analysis capabilities. Utilization of light-pens, menus, function keys, and multiple view ports with windowing, zooming, and moving capabilities, together with support for additional graphics as well as system and application utilities, create a functionally rich and user-friendly interface to the system. In addition to its graphics interface, there is also a command language interface available for GMSOLID, and the two interfaces are compatible, thereby permitting users to set up and schedule complex background jobs through the graphic interface and continue with other interactive tasks or terminate the interactive session and have the results ready by the next session [BOY82].

Internally, GMSOLID keeps two concurrent representations of design objects: a CSG tree structure and a boundary representation [BOY79b, BOY82]. Availability of a boundary representation makes it possible to also cater to applications where explicit representations of faces, edges, and vertices are required (in addition to those for which CSG trees are more suitable). A boundary evaluator subsystem is responsible for generation of the boundary representation of solids composed of other solids through CSG operations. This process involves creation of new geometric entities, i.e. surfaces, edges, and vertices, removal of old ones invalidated by a later operation, and merging of coincident entities into one. Because this is potentially a time-consuming process, GMSOLID makes it possible to delay boundary evaluation until a user attempts an operation which requires an up-to-date boundary representation, and then evaluation can be carried out in the background, if so desired, permitting the interactive session to continue.

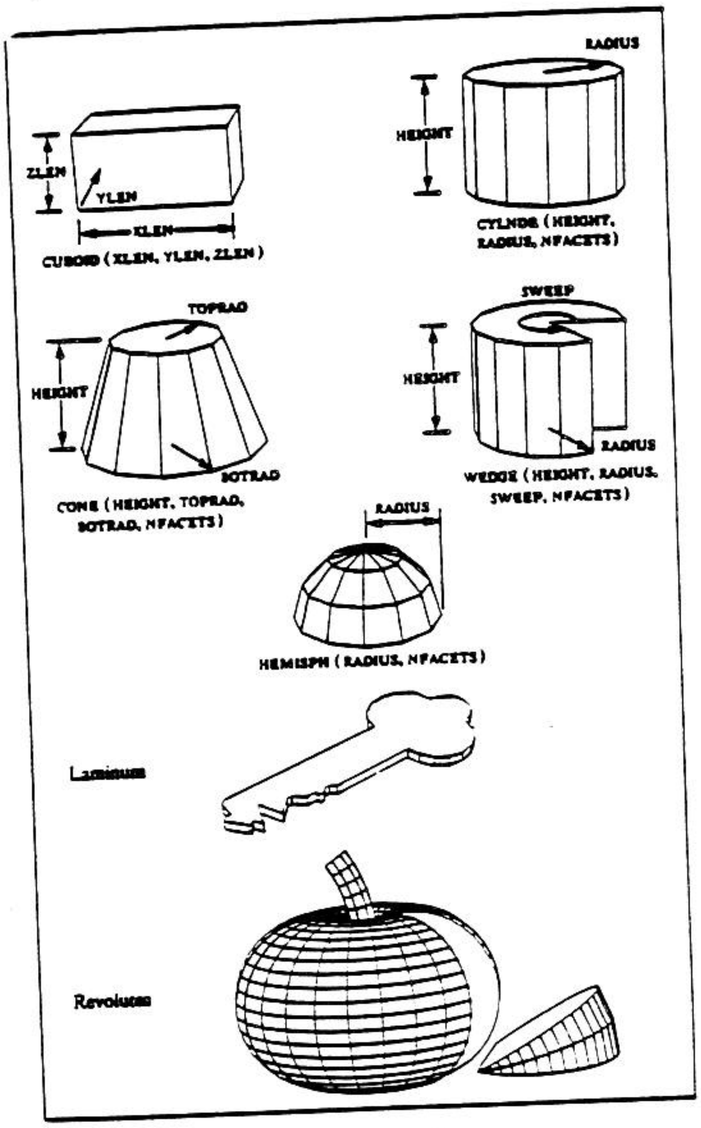
Surfaces in GMSOLID are quadratic, i.e. second-order polynomials in x, y, and z, and its set of basic primitive solids consists of blocks, cylinders, cones, and spheres. An interface to CADANCE permits some of the facilities which already exist in that system to be used through

GMSOLID. For example, a user can define a planar closed contour using lines and arcs in CADANCE and then request GMSOLID to create a solid by "extrusion", i.e. translational sweeping, of this contour [BOY79b, BOY82]. This interface also allows access to the existing CADANCE data base.

3.4.3 GDP/GRIN (IBMSolid)

GDP, for Geometric Design Processor, is a general volumetric modeling system developed at the IBM T. J. Watson Research Center. GDP grew out of research on high-level languages for programming computer-controlled robotic or mechanical assembly at this center, the result of which was definition of a language called AUTOPASS (for AUTOmated Parts Assembly System) [MYE82, WES80a]. In the course of this work, it became evident that solid modeling played an important role in a robot's awareness of its physical environment.

In an earlier work [GRO76], a package of PL/I procedures was developed that allows hierarchical definition of geometric objects in terms of PL/I procedures with very general use of variables. This differs from the underlying approach of PADL, for example, in that in this approach the nodes in the hierarchy defining a part are themselves procedures, whereas a PADL part definition is recorded entirely as a data structure. Object definitions are, therefore, not "static" as in PADL and are interpretively executed, thus permitting association of generalized semantics to object description trees. Together with heavy use of variables, this provides a means for modifying the semantics associated with an object without modifying its procedural description.



RADIUS
HEIGHT
CYLNDR (HEIGHT, RADIUS, NFACETS)
ZLEN
YLEN
XLEN
CUBOID (XLEN, YLEN, ZLEN)
TOPRAD
HEIGHT
BOTRAD
CONE (HEIGHT, TOPRAD, BOTRAD, NFACETS)
SWEEP
HEIGHT
RADIUS
WEDGE (HEIGHT, RADIUS, SWEEP, NFACETS)
RADIUS
HEMISPH (RADIUS, NFACETS)
Laminum
Revolutes

From: [WES80a]

Figure III.23 - Primitive Solids in GDP

GDP uses a procedural representation scheme based on the work in [GRO76] to create initial geometric descriptions. Users design objects by combining primitive solids using CSG-like operations [WES80a]. This is done by making PL/I calls to four generic subprograms which permit building positive (solid) or negative (hole) geometric entities and creation of parts and aggregates. The supplied set of primitive solids includes cuboids (blocks), cylinders, wedges, cones, hemispheres, laminae (flat, sheet metal type of objects), and revolutes (volumes of revolution). See Figure III.23. In addition, users can define and use their own solids by supplying appropriate object procedures, which may in turn call other user-supplied object procedures or use the system's primitive solids. Solids can also be created as deformation of other solids through non-homogeneous three-dimensional scaling, e.g. an egg-shaped solid can be defined by scaling a sphere using three different scale factors along x, y, and z axes. All solids in GDP are modeled by polyhedral approximation. Since fineness of approximation, i.e. number of planar faces, is simply one of the variables associated with the definition of an object, it can be changed at any time before a solid is merged[6], without requiring a structural modification to its definition tree. This permits interactive users to begin with a low facet number, resulting in crude approximations but efficient computation and faster response, and to request a higher number of polyhedral faces only when required.

GRIN [FIT81a], for GRaphic INput subsystem, is an interactive graphic interface to GDP that runs under VM/CMS and uses an IBM 3277 alphanumeric terminal with Tektronix storage tube Graphic Attachment. Future expansions of GRIN will allow use of other graphic terminals, specifically, IBM 3250 type vector refresh scopes. In addition to entity creation functions, GRIN provides convenient facilities for display and windowing functions such as translation, rotation, zooming, hidden-line removal, and multiple views, together with graphic editing

6 Merging of solids occurs when they are combined to create another solid. The only way to change the facet number of a (component of a) merged solid is to "unmerge", change the facet number, and merge back.

operations such as those for replicating, mirroring, copying, rotating, moving, and naming of objects. Entity creation functions in GDP/GRIN allow entry of points, lines, and arcs (approximated by line segments) to serve as references for creation and positioning of solids, permit instancing of primitive solids, definition of solids through translational and rotational sweeping of planar polygons, and, finally, fetching of object definitions from disk libraries [FIT82]. A dozen tree editing functions permit traversal and editing of tree structures that define objects.

If a light-pen is available, GDP/GRIN functions can be invoked by selection of menu items. The same host of facilities are also available through a simple set of commands that can be typed on a keyboard. GRIN currently has an interface to CADAM which can be used to transfer geometric definitions to CADAM for additional drafting-type processing, such as posting of dimensions. A command level macro facility (with limited capabilities) and an on-line help feature make GRIN a flexible and user-friendly interactive interface.

While the underlying approach in GDP permits inclusion and treatment of general attributes of objects, GDP/GRIN presently focuses only on geometric and graphic properties of objects.

CHAPTER IV

An Operational Approach to Solid Modeling

This chapter pursues an operational approach to geometric solid modeling and proposes a manufacturing-oriented modeling scheme, called DSG, as the skeleton for a manufacturing-oriented design language suitable for an integrated CAD/CAM system.

Section IV.1 establishes the underlying mathematical formalism for DSG. It begins by a closer re-examination of CSG as a logical model for mathematical representation of solid objects and, in doing so, unveils a number of inconsistencies. The remainder of this section develops an alternative coherent mathematical scheme, called Realizable Shape Calculus (RSC), for consistent representation of physical solids.

Section IV.2 justifies an operational, manufacturing-oriented approach to design and definition of mechanical parts and defines and discusses major features of DSG in terms of RSC. Although this discussion assumes RSC as its underlying mathematical model for representation of solid objects, the methodology and concepts proposed and advocated by DSG are independent of this formalism and can be applied to implement manufacturing-oriented design languages based on other geometric modeling systems as well. However, implementation of DSG concepts on a modeling scheme other than RSC (e.g. CSG) could possibly be restricted by its more limited mathematical expressiveness.

Finally, Section IV.3 provides several examples of DSG part definition and in conclusion, Section IV.4 discusses the benefits and implications of our manufacturing-oriented methodology and exposes its potentials in some of the areas which are topics for further research.

The notation used in this chapter is consistent with Appendix A, which covers certain fundamental definitions and topological properties of Euclidean three-dimensional space, E^3. The symbol $\emptyset$ represents the empty set. If X is a set then X', $\overline{X}$, $\beta(X)$, int(x), and ext(X) are, respectively, its complement, closure, boundary, interior, and exterior.

1 Formal Models of Solid objects

The most significant characteristic of a real solid object is its geometric shape. Therefore, it is quite natural to model solid objects as subsets of E^3 whose geometric properties correspond, in an abstract and idealized fashion, to those of the modeled objects. Not every subset of E^3 is a reasonable model for a real solid. It is essential to establish a definition for "real solid objects", i.e. determine the set of properties and conditions with which we expect to characterize the notion of "real-world solidity", before a suitable formal model for solid objects can be developed. One such set of properties is discussed in [REQ80c] and [REQ77c] and is rephrased below:

1- Rigidity

Rigidity refers to the property of invariance of shape under translation and rotation.

2- Homogeneous three-dimensionality

A solid must occupy a "volume", i.e. it must have an interior, and it cannot have isolated or "dangling" boundary segments or "infinitely thin cracks".

3- Finiteness

A solid must occupy a finite portion of space.

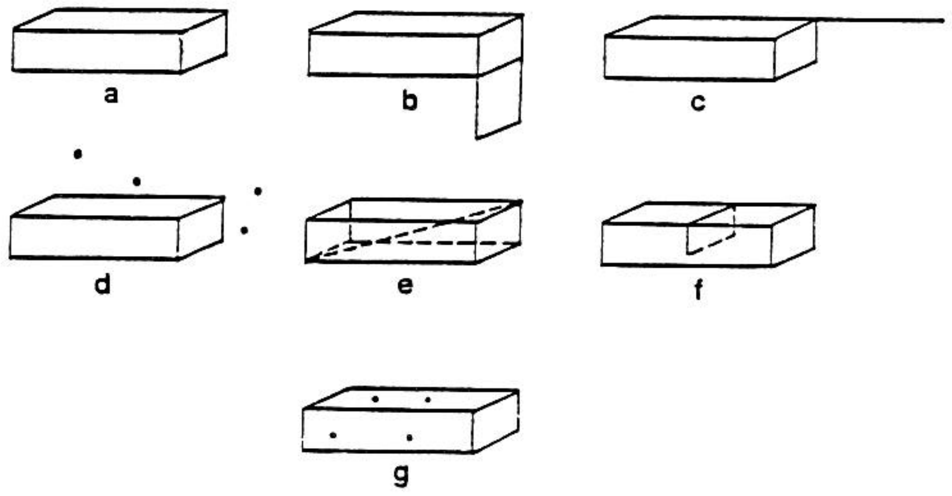
4- Closure under physical operations

A set of physically conceivable operations on solids must produce solid objects; e.g. relocation, addition, and removal of solids must yield solids.

5- Boundary determinism

The boundary of a solid object must determine unambiguously its "inside" and "outside".

In addition to capturing the above set of properties, a modeling scheme for representation of solids, just like any other computer model, must be capable of associating a finite description (symbol) to every representable entity. It is proposed in [REQ80c] and [REQ77c] that subsets of E^3 which satisfy these conditions are suitable models for real solids.



(a) Solid block; (b) Solid blocks with dangling face; (c) Block with dangling edge; (d) Block with isolated points; (e) and (f) Blocks with infinitely thin cracks (a void line and a void surface); (g) Block with infinitely thin holes (void isolated points)

Figure IV.1 - Legal and Illegal Set Models for Real Solids

Homogeneous three-dimensionality means that subsets of E^3 with isolated points, dangling edges and/or faces, or infinitely thin cracks and/or holes are unacceptable. See Figure IV.1. Disconnected regions, however, are permitted as "assemblies". The rigidity condition guarantees closure of solids under translation/rotation. This condition also excludes fluids, i.e. liquids, gas, and powdered solids, because in general, they fail to preserve their geometric shapes after relocation. Another common connotation of rigidity is the invariance of shape in time. However, requiring this condition can lead to problems due to decay of solids, chemical reactions (e.g. rusting), and gravitational deformations which we do not intend to consider in geometric modeling. Constructive Solid Geometry (CSG) shows that the shape of any three-dimensional solid piece can be represented by (regularized) union, intersection, and difference of a suitably rich set of primitive solids (see Section III.2.2). These operations, therefore, typify a minimum set of operations besides translation/rotation, under which a solid model should be closed. Boundary determinism prevents such unreal surfaces as a Klein bottle (Figure III.13) and requires well-behaved boundaries with realizable mathematical properties, e.g. bounded area in a bounded domain.

1.1 R-sets

In [REQ77c] it is argued that a suitable model for real solids is the set of all (congruence classes of) subsets of E^3 which are bounded, closed, regular, and semianalytic. Such sets are called r-sets. This section examines the implications of using r-sets to model solid objects, and Section IV.1.2 shows their shortcomings. In order to overcomes these shortcomings, a new mathematical model for representation of physical solids will then be introduced in Section IV.1.3. This representation together with a suitable set of operations discussed in Section IV.1.4, comprise a calculus of realizable shapes, a modeling scheme in the same spirit as CSG, but with a wider range of application.

We begin with a reexamination of the conditions imposed on r-sets. The boundedness condition is, of course, an important requirement that corresponds to the finiteness property of real solids. There are, however, certain special cases where it is meaningful to consider "unbounded solids". In CSG, the more general term "half-space" is used to refer to "solids" which may be unbounded [REQ77b].

The boundary of a solid object separates the object from its environment. Thus, it is natural to model the boundary of a solid object as the topological boundary of the subset of E^3 that represents that object. A set is <u>closed</u> if it contains its own boundary (see Appendix A). A subset of E^3, say X, is called a closed-regular[1] set if it is equal to the closure of its interior, that is if $X = \overline{\mathrm{int}(X)}$. The formal condition of closed-regularity can be viewed as a relationship between a set and its boundary that to some extent reflects our intuition about the relationship between an object and its boundary. (For more discussion on this issue, see Section IV.1.3.) Closed regularity excludes subsets of E^3, such as those in Figures IV.1.b, c, and d, where (a portion of) the boundary does <u>not</u> separate "inside" from "outside".

Boundedness and closed-regularity are not sufficient to guarantee correspondence of subsets of E^3 to real objects. For instance, with no

1 In solid modeling literature, the term "regular" usually implies closed-regular, but for clarity, we will avoid such abbreviation.

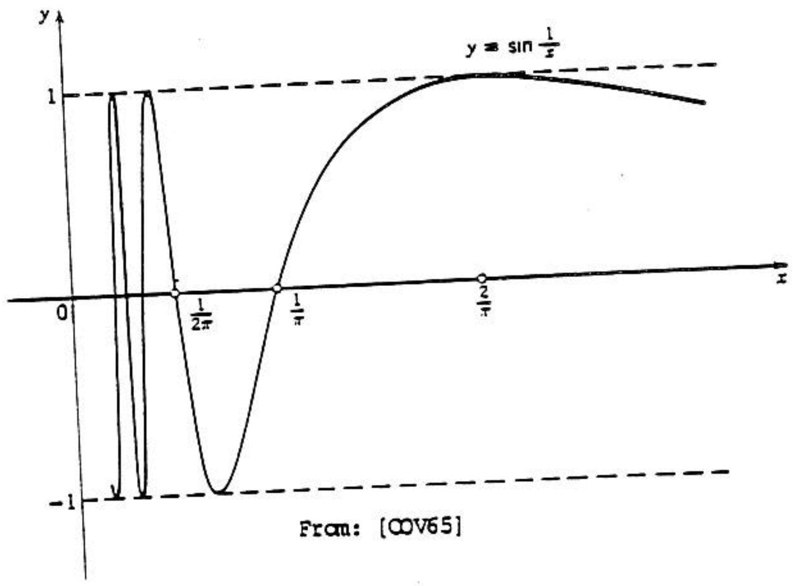


Figure IV.2 - Infinite Oscillation

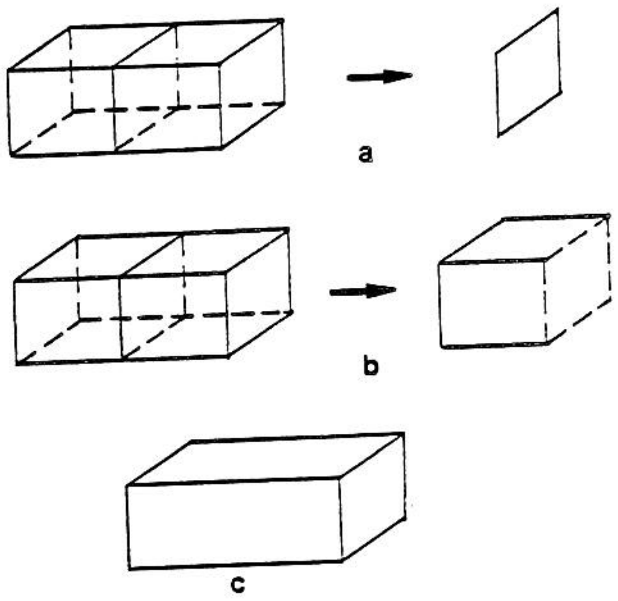
further restrictions, the boundary of a closed-regular, bounded set can exhibit unacceptable properties such as an infinite number of non-dampening oscillations in a bounded domain, as seen in Figure IV.2. Such "wild" boundary behavior is an impossible characteristic for a real solid. (For example, consider the "surface" resulting from extrusion of the graph in Figure IV.2. The area of this surface is unbounded in any bounded neighborhood of x=0.)

In practice, restrictions on the shape and type of surfaces that can be handled by a system (e.g. using polyhedral or polynomial approximation) preclude such ill-behaved models. For a generalized solid modeling scheme, however, a formal definition of acceptable shapes is required. To prevent pathological behavior of boundaries while supporting a wide class of realizable shapes, [REQ77c] defines boundary well-behavedness through the notion of semianalyticity. A function

$F: E^3 \rightarrow E^3$ is called analytic in a domain if it can be expanded into a convergent power series about every point of the domain. A set of points in E^3 is called analytic if it can be expressed as $\{ (x,y,z) \mid F(x,y,z) \leq 0\}$, where F is an analytic function and x, y, and z are the coordinates of a point. A set is called semianalytic if it can be expressed as a finite boolean combination (through union, intersection, and complementation) of analytic sets. The semianalyticity condition of r-sets prevents unrealizable boundaries such as the one in Figure IV.2.

In addition to a formal representation for solid objects, a solid modeling scheme must be able to express various kinds of relationships between objects' shapes as they undergo certain physical processes of interest. Typically, one is interested in combining shapes in such a way as to represent the result of processes like "gluing", material removal, and assembly, and to solve such shape-related problems as interference checking and adjacency.

An obvious set of mathematical operators suitable for such a task consists of the set operations of union, intersection, difference, and complementation. However, r-sets are not closed under these operations and unrealizable models can result from their application, as seen in Figure IV.3. A modified version of the set operations that preserve closed-regularity and, of course, semianalyticity, are the regularized set operations described in [REQ77c], which are fundamental to CSG. In CSG, the regularized union ($\cup^*$), intersection ($\cap^*$), and complement (c^*) of two sets X and Y are defined as $X \cup^* Y = \overline{\text{int}(X \cup Y)}$,



(a) Intersection of two touching blocks yields a plane

(b) Difference of two touching blocks yields a non-closed block

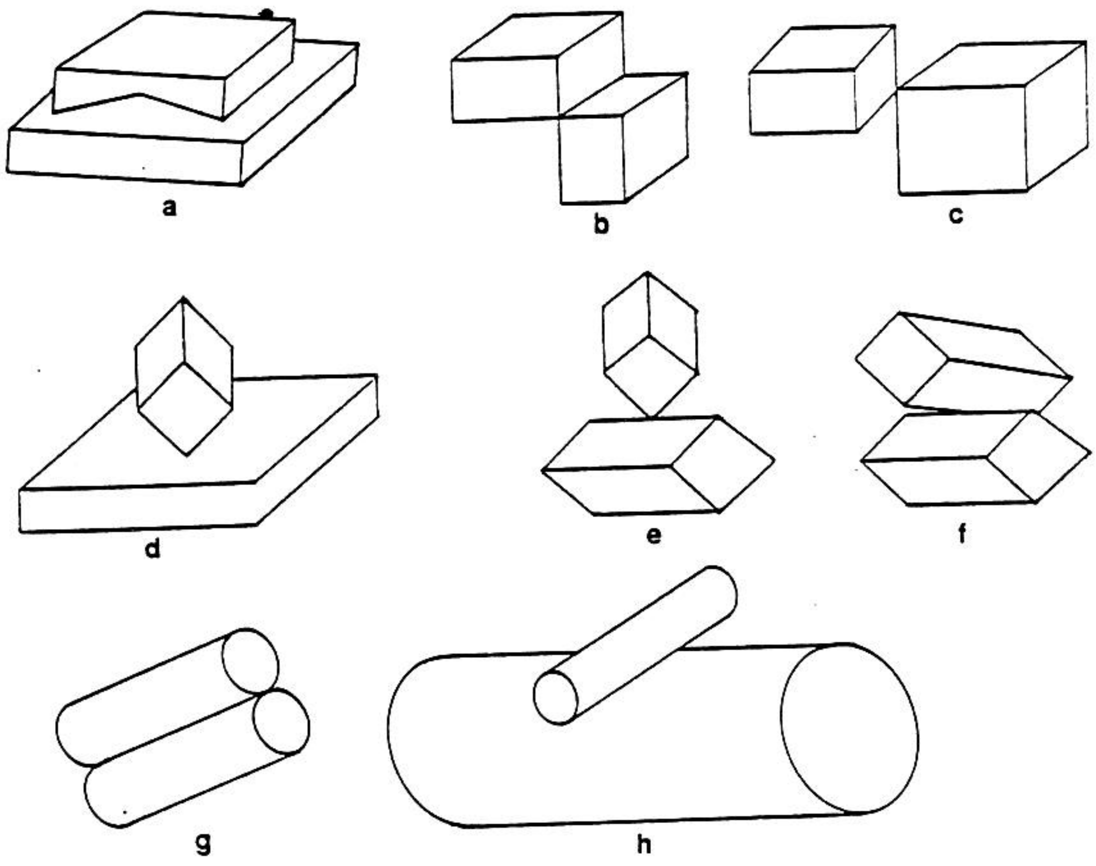
(c) Complement of an r-set is not bounded

Figure IV.3 - Inadequacies of Set Operations

$X \cap^{*} Y = \overline{\text{int}(X \cap Y)}$, and $c^{*} X = \overline{\text{int}(X')}$. Regularized difference $(-^{*})$ can be defined analogously, or through regularized intersection and complement, i.e. $X -^{*} Y = X \cap^{*} (c^{*} Y)$ (see Figure IV.11 for a depiction of the effect of these operations on r-sets). The class of closed-regular semianalytic sets (half-spaces) is closed under these operations [REQ77b]. In particular, r-sets are also closed under regularized union, intersection, and difference, but regularized complement of an r-set is not an r-set (because it is not bounded). This simple set of operations is mathematically expressive enough to permit composition of a wide variety of shapes, given a small number of elementary r-sets called primitive solids (see Figures III.15 and IV.13, for example), and if we believe that all r-sets represent realizable shapes, closure of r-sets under these operations guarantees that the result of any combination of realizable solids is in fact a realizable solid.

1.2 Limitations of r-sets

The r-set model of solid objects includes sets shown in Figure IV.4 which represent different special cases where the result of a CSG operation, e.g. (regularized) union, on r-sets yields an r-set whose boundary is not a two-dimensional manifold.[2]



(a) Edge-face (b) Edge-edge (c) Vertex-vertex (d) Vertex-face (e) Vertex-edge (f) Edge-edge (g) Face-face (h) Face-face

Figure IV.4 - Non-manifold R-sets

2 A subset X of E^3 is a two-dimensional manifold if every point in X has an open neighborhood homeomorphic to an open disc and satisfies another condition (too technical for our purposes to mention).

Strictly speaking, such non-manifold sets in CSG, represent unrealizable objects, or perhaps, physically unrealizable arrangements of realizable objects. In either case, their unrealizability is due to existence of an "infinitely thin solid region", e.g. a solid edge or vertex. Nevertheless, such r-sets are intentionally permitted in CSG, because they can be interpreted as somewhat reasonable models for assemblies of objects [REQ77c]. Figure IV.4.a, for example, can be accepted as a model for two parts, one resting on the top of the other; with proper modifications, similar meaningful interpretations can also be associated with other sets in Figure IV.4.

Accepting these r-sets as assemblies means that they should be regarded as composed of disconnected pieces. This interpretation implies that vertices and edges with non-manifold neighborhoods should somehow be considered as not belonging to objects. The tendency to exclude these anomalous boundary segments is further justified by the fact that in CSG, solid regions with such "abnormal" intersections are formally considered to be non-interfering; for example, the regularized intersection of every pair of r-sets in Figure IV.4 is empty. However, if vertices and edges with non-manifold neighborhoods were to be excluded, then the sets in Figure IV.4 representing individual pieces would be non-closed sets which are not r-sets, by definition. This shows that the pairs of r-sets in Figure IV.4 represent ambiguous cases to which no fully consistent interpretation can be associated in CSG.

The same inconsistency also becomes apparent by observing that, in the real world, when two objects, say A and B, do not interfere, their aggregate (union) consists of at least two disconnected regions (in the sense that there is no path, i.e. curve, fully contained in their union, i.e. A U B, that can connect a given pair of points not in the same region). Pairs of r-sets represented in Figure IV.4 are examples where CSG and r-sets fail to reflect this real-world property of disconnectedness. To underline the significance of this issue, consider, for example, an algorithm for computation of the effect of exerting a force f to the top part of the object in Figure IV.4.a, opposing a gravitational field g for t seconds. The result of this algorithm would, no doubt, depend on its programmer's interpretation of this geometric model, and is therefore formally ambiguous.

Although it is possible to introduce a rule which would give the "correct" interpretation for such non-manifold models in CSG (i.e. implement an algorithm which would recognize them as disconnected), abiding by this rule (i.e. invoking the algorithm in every relevant occasion) would necessarily be a completely voluntary compliance on the part of users (algorithms as well as humans) in such a scheme. The reason is that this rule is neither enforced nor preserved by the operations within the modeling scheme itself. A "good" logical model (or abstract data type) cannot afford to rely on its users to comply with its interpretation or integrity rules; it must enforce them.

A second problem associated with the r-set model of solids, shared also by any solid modeling scheme based on closed sets, is that certain assemblies of objects cannot be represented consistently. For example, Figure IV.5.a shows (the front view of) a thin block resting on the top of a larger block with the same depth. The sets represented in this figure comprise a legitimate model for an assembly of two disconnected

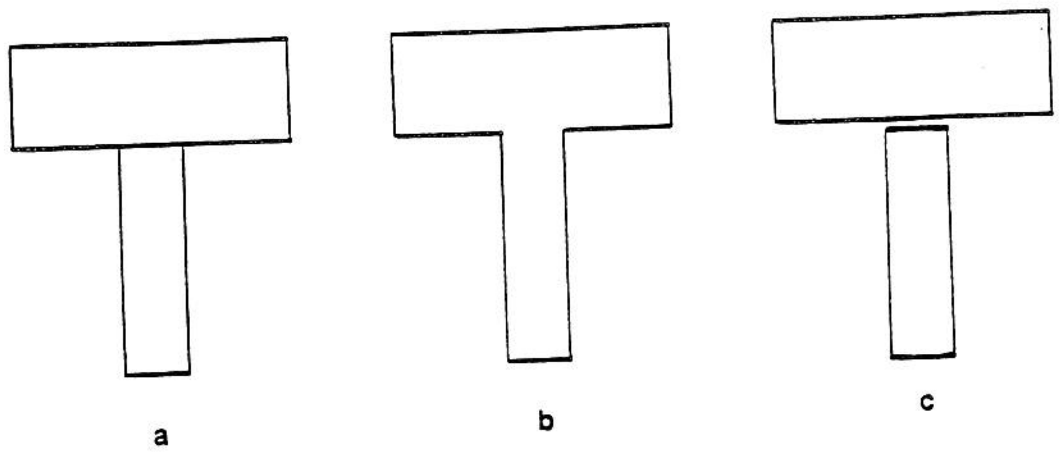


Figure IV.5 - Assembly vs. Part

pieces; this model is at least as legitimate as the ones in Figure IV.4, which are permitted in CSG and are meant to be interpreted as assemblies. However, the "set" shown in Figure IV.5.a is not an r-set at all (because this representation cannot be that of a closed set)[3]. The r-set model of this "assembly" is shown in Figure IV.5.b, which in fact, does not represent an assembly! In the real world, the assembly

3 A few words on depiction of set-models of solids in general, is in order. Drawings such as those in Figure IV.5 are graphic representations of sets, which in turn, we interpret as models for solid objects. The common-sense convention in line-drawing representation of sets, which is observed throughout this dissertation, is that lines (or curves) depict the loci of points with certain distinctive topological attributes associated with boundary points of sets. The loci of such points (and only such points) are depicted as lines and curves, regardless of whether or not they belong to the sets in question. Therefore, whereas Figures IV.5.b and c can indeed be depicting closed sets, Figure IV.5.a, cannot be a depiction of a closed set, because it includes a line segments that with the closed-set interpretation of this drawing, becomes the locus of non-boundary points, i.e. should not have been drawn.

of Figure IV.5.a would have certain properties quite different from those of the very similar object with the same overall shape, made out of a single connected piece (or several pieces "glued" together), shown in Figure IV.5.b. (See also the example shown in Figure IV.7, discussed in the following section.)

A "patchwork" measure that would allow this type of adjacent assemblies to be representable in a modeling scheme based on closed sets, is to represent adjacency by a distance smaller than a given threshold d>0. With such a measure, the assembly of Figure IV.5.a would be modeled as shown in Figure IV.5.c. Unfortunately, this patchwork solution suffers from two major flaws:

1- For every given value for the threshold distance d, below which two close objects are to be considered "touching" in a modeling scheme, one can always construct counterexamples involving objects with yet smaller dimensions which would thus not be representable in that scheme. A remedy is to let the threshold d be variable rather than a preselected constant. Although it is conceivable to devise a modeling scheme wherein the actual value of d is context-dependent (e.g. takes into account the relative sizes of neighboring objects), such measures would make the scheme too complex, especially when the issue of tolerances is also considered. Besides, the semantics of an "assembly" with such a context-sensitive threshold is inherently ambiguous when "context" is subsequently changed, e.g. two objects are cut in size after they are defined to be "touching".

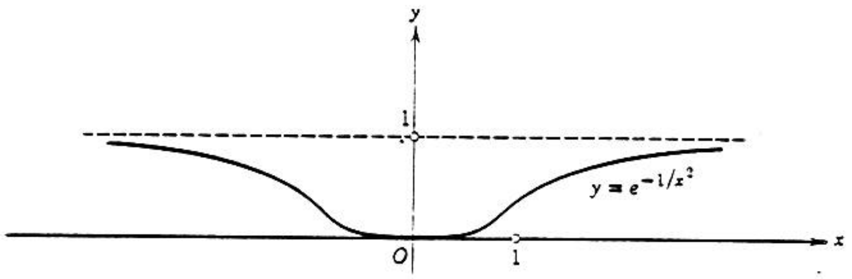
2- Given that the sets of Figure IV.4.b, for example, comprise (albeit, still arguably) a model for an assembly, i.e. for two disconnected touching blocks, the fact that a slight translation of the top block to the right would produce a connected piece, instead of an assembly of a block partially resting on the other (represented through any "patchwork" solution suggested above, for example), is both inconsistent and counterintuitive.

We thus conclude that the important class of touching assemblies cannot be represented consistently in any solid modeling scheme based on closed sets.

A third problem with the r-set model of solids is that it excludes certain reasonable boundaries due to the semianalyticity condition. Recall from Section IV.1.1 that semianalyticity means that the boundary of a set can be divided into a finite number of segments, each of which is analytic in its domain, i.e. can be expanded into a convergent power series. Recall also that the reason for imposing this condition was to permit only well-behaved boundaries, meaning those that are continuous and "smooth". In order for a function to be expandable into a power series, all of its derivatives must necessarily exist at the point(s) in question, i.e. it must be "smooth" and continuous. However, existence (and continuity) of all derivatives is not sufficient for expandability of a function [COU65]. Therefore, by requiring the stronger condition of expandability into power series, r-sets exclude certain reasonable shapes. A classic example of a function with continuous derivatives which is not expandable into a power series is the function f, defined as:

$$f(x) = \begin{cases} e^{-1/x^2} & \text{if } x \neq 0 \\ 0 & \text{if } x = 0 \end{cases}$$

The graph of y=f(x) is shown in Figure IV.6.



From: [COU65]

Figure IV.6 - A Non-expandable Continuous Function

Function f and all its derivatives are continuous in every interval, even at x=0; however, because all of its derivatives vanish at X=0, this function is not expandable into a power series [COU65]. As seen in Figure IV.6, this function is quite reasonably smooth and well behaved everywhere, yet (the three-dimensional analog of) this function is disallowed as the boundary of an r-set.

Some of the problems discussed above, arise from the fact that r-sets are closed, i.e. contain their own boundaries. To resolve these problems, it is then natural to seek an alternative to closed sets where boundary points are excluded from a set. Such an alternative does exist: A set is <u>open</u> if it does not contain any of its boundary points. (Note that a non-closed set is not necessarily open; see Appendix A.) It is indeed tempting to attempt to develop a formal model for solid objects quite similar to r-sets but based on open sets; in other words, to model physical solids by open-regular, semianalytic sets. For the sake of argument, let us consider such a model and study its characteristics.

The concept of open-regularity can be defined analogously after closed-regularity: A subset of E^3, namely X, is called open-regular if it is equal to the interior of its closure, i.e. if $X = int(\bar{X})$. Open-regular sets are the duals of closed-regular sets and essentially share many of their interesting properties, e.g. the relationship between a set and its boundary which prevents dangling boundary segments and is of particular interest to us. For example, the representations in Figure IV.1 which were not valid as closed-regular sets, are not acceptable as open-regular sets either. Imposition of the semianalyticity condition on open-regular sets produces open half-spaces. The "open version" of r-sets, which we call ro-sets, can be defined after r-sets as bounded, open-regular, semianalytic sets. Likewise, one can define an "open version" of the regularized set operations under which open half-spaces and (excluding regularized complement) ro-sets are closed. This development results in a modeling scheme analogous to CSG but based on open sets, to which we refer as OCSG.

OCSG, however, suffers from the same problems as CSG does, and ro-sets are only slightly better models than r-sets for solid objects. The problem of ambiguous interpretation of non-manifold models with r-sets is resolved in the ro-set model of solids: Considered as ro-sets, the representations shown in Figure IV.4 have the unique interpretation of being disconnected and thus are consistent models for assemblies of objects. (Note that, although the boundaries of these sets are still non-manifold boundaries, the sets themselves are manifolds since they do not include these boundaries.) On the other hand, the other two problems with r-sets, mentioned above, still remain with ro-sets. Clearly, ro-sets and r-sets are both capable of representing the same class of three-dimensional shapes since they are both based on analytic functions. Therefore, (the three-dimensional analog of) shapes such as in Figure IV.6 are just as non-representable with ro-sets as with r-sets.

Like r-sets, ro-sets too cannot consistently represent touching assemblies. Although it is possible to consistently represent the class of touching assemblies using open sets (s-sets defined in Section IV.1.3, for instance, are capable of doing so), open-regular sets do not posses this property due to their regularity. For example, Figure IV.5.a may be a depiction of two touching open sets and thus can be interpreted (uniquely) as a model for a touching assembly. However, the "set" in Figure IV.5.a is not open-regular and therefore is not an ro-set[4]. Interestingly, the ro-set model of the assembly in question is very similar to its r-set model and, again, is represented by Figure IV.5.b. The same "patchwork" measures as discussed earlier for r-sets, can be used with ro-sets in order to permit modeling of touching assemblies, but again, they lead to schemes that suffer from the same shortcomings.

4 See footnote 3 in page 105 regarding graphic representation of sets, and substitute "ro-set" for "closed set" in the text of that footnote.

We summarize the results of this section by concluding that:

1- Closed sets cannot be used to consistently model assemblies of objects.

2- Regularity (closed or open) is too restrictive a condition to insure "separating boundaries" with no dangling segments or "infinitely thin cracks", because it disallows consistent modeling of assemblies.

3- Analytic functions are too restrictive for mathematical definition of realizable shapes.

1.3 S-sets

In this section we present an alternative model for mathematical representation of physically-realizable solid objects that covers a wider class of reasonable shapes than do r-sets, and at the same time, lack their inconsistencies. We refer to the subsets of E^3 that fit our proposed definition as s-sets (for "solid-sets"). Not only s-sets are more reasonable models than r-sets for mechanical parts, they are also consistent models for assemblies, as we show in this section. Taking advantage of this point, Section IV.1.4 presents a CSG-like solid modeling scheme based on s-sets which in addition to operations equivalent to the regularized set operations, includes a new "assembly operator". Although s-sets are referred to throughout the present section, formal definition of s-sets, Definition 1.4, appears at the end of this section. However, a first reading of the following discussions that develop concepts leading to Definition 1.4, requires no more detailed understanding of s-sets than the above introduction.

We take the position that the boundary of an object is incidental rather than existential, i.e. it is a side-effect of the existence of the object rather than an independent physical manifestation. Furthermore, the notion of "physical boundary" as well as the definite boundary of an object according to any reasonable definition of the term, both become quite "fuzzy" at microscopic scales. Hence, we contend that in macroscopic modeling of objects' shapes, one should regard "boundary" as an abstract concept rather than a physical entity, whose ideal geometric shape and position can only be approximated to a given precision. In other words, one should treat boundary as a non-entity that asymptotically delimits an object. We thus model physical solid objects as open sets of material points bounded by surfaces, edges, and vertices which they do not own (i.e. contain), and which separate them from another open set of void[5] points in space. The two immediate consequences of this view are:

1- Surfaces, edges, and vertices are abstract delimiters (i.e. without real counterpart) that separate two inherently analogous regions of space, namely, an occupied and a void region. Such abstract entities are thus meaningless (therefore do not exist) out of this context.

5 "Void" only in the sense that they belong to an object's spatial environment, rather than to the object itself.

2-Since objects do not own their boundaries, they cannot "share" them either, e.g. two blocks can be moved arbitrarily close to each other, but they cannot be connected at a corner, or at an edge (or even "at a surface", as discussed in the example of Figure IV.7.c, below).

Because bounded open sets always have boundaries which are not contained in the sets, the integrity of an object as an independent identifiable entity, is always preserved when it is modeled by an open set; the same is not true with closed sets, as evidenced by examples in Figure IV.5 and in Figure IV.7, below. This property of open sets allows us to model the important concept of physical adjacency, without which a consistent model of assemblies of objects is not possible.

Since exclusive use of open sets precludes connectivity (of sets) through boundary sharing, our proposed modeling scheme (based on s-sets), can use this topological property to model physical adjacency. Two open sets with a common boundary segment are theoretically at zero distance from one another, but clearly, real objects cannot be placed at "zero distance" apart; no matter how close two adjacent real objects are, there is always a positive distance, the value of which is very context-dependent, between their boundaries. Nevertheless, it is both unreasonable and impractical to deal with the exact distances between physically adjacent objects in modeling of touching assemblies because, unless such distances play a functional role in the assembly (which they virtually never do[6]), any definite value would be equally irrelevant and

6 An example of where such distances can play a functional role, is in integrated circuits where "adjacent layers" are separated by

misleading. The undeterministic concept of physical adjacency, with all its formal ambiguity and context sensitivity, seems to be an important notion in the real world that within a given application context, is virtually never ambiguous in practice. To properly model this notion, physical adjacency, we need to model the non-deterministic concept of the distance between two objects which are "as close to each other as possible". The most appropriate "symbol" in a modeling scheme for representation of this "indefinitely small" value is zero, because, unlike other real numbers, this symbol cannot be taken literally to represent a distance between two objects in the real world. Hence, we adopt the following definition:

1.1 DEFINITION: Physical Adjacency

Sets with common boundary segments, i.e. zero distance apart, are models for assemblies of objects which are "as close to each other as possible", within the proper application context. □

Although the above definition is far from being formal, its underlying logic is so intuitively clear and naturally obvious that it has always been used (indirectly and even less formally) in virtually every type of solid representation technique from engineering drawings to present solid modeling systems. On the other hand, no geometric solid modeling scheme has offered a formal basis for consistent inclusion and treatment of assemblies as objects. For example, although most CSG-based systems such as PADL [PAD74], GMSOLID [BOY80, BOY82], and

functionally significant distances. But, it is doubtful whether inclusion of such cases in the category of mechanical assemblies can serve any purpose.

GDP/GRIN [WES80a, FIT81a], provide operations and commands for building assemblies of objects, their underlying formalism, CSG, in itself is not capable of dealing with assemblies in the same manner (see Section IV.1.2).

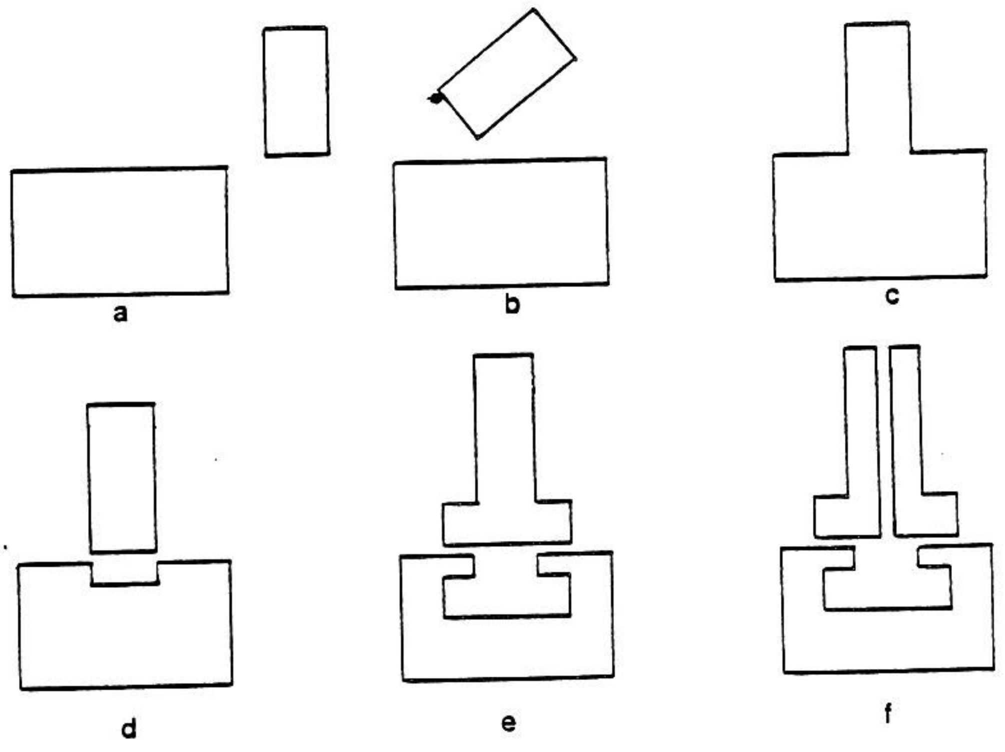


Figure IV.7 - Touching Assemblies Modeled by Closed Sets

To make this point clear, consider as an example, the two blocks depicted in Figure IV.7.a within the interpretation context of CSG. Using (regularized) union as the assembly operator in CSG, Figure IV.7.a shows an assembly of the two blocks at a considerable distance apart. Figure IV.7.b shows them at a closer distance and at a different relative orientation. Figure IV.7.c represents the same two blocks assembled "at zero distance" from each other, i.e. physically adjacent. Note that the physical integrity of the two objects is violated by adjacency, i.e. the "two" blocks are now one object in Figure IV.7.c. On the other hand, in PADL or GDP/GRIN, whereas representation of assemblies of far apart objects (as those in Figures IV.7.a and b) are identical to their CSG representations, touching assemblies, such as a block resting on the top of another, have a different representation (as in Figure IV.8.a) than their CSG models (Figure IV.7.c). In theory, then, the touching assemblies defined in PADL or GDP/GRIN, cannot be operated on by CSG operators because, technically, they are not half-spaces, whereas the CSG models for the same assemblies are not recognized as such by these CSG-based systems.

Furthermore, consider the transition of the two blocks in Figure IV.7.b to the configuration of Figure IV.7.c in CSG. One can argue that objects' shapes are apparently changed as a result of translation/rotation alone (the "two" blocks in Figure IV.7.c are still "non-interfering" because their regularized intersection is empty). This example, then, shows that either CSG violates the rigidity property of real solids (see Section IV.1), or that it proposes that two objects, while maintaining their separate identities, can be connected at exactly a surface, without any mutual interference (e.g. one can still "slide" on the other). Finally, note that Figure IV.7.c is also the CSG model for, among other things, the sets of objects depicted in Figures IV.7.d, e, and f, when "assembled" properly.

The above problems can be avoided in a solid modeling scheme based on open sets because the boundary of an open set can maintain its independence and integrity of shape through every modeling operation, other than the ones that explicitly modify its boundary. Such

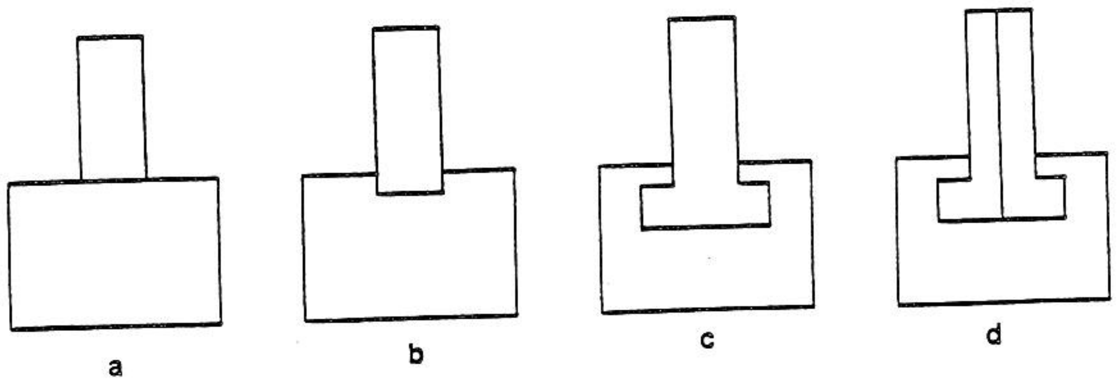


Figure IV.8 - Touching Assemblies Modeled by Open Sets

operations, in turn, can be carefully devised so as to represent manufacturing processes that modify an object's shape. (The OCSG scheme discussed in Section IV.1.2 provides a counter-example where this correspondence of modeling operations to manufacturing processes is not observed.) The four different assemblies discussed above, all of which would be represented by the set shown in Figure IV.7.c in a closed-set based modeling scheme like CSG, are depicted in Figure IV.8 as modeled by (non-regular) open sets[8]. (Note that except for the open vs. closed difference, the sets in Figure IV.8 <u>are</u> representations for the same assemblies in CSG-based systems.)

The intention of Definition 1.1 was to express the condition under which two solid objects are considered physically adjacent (i.e. touching). This same intention is rephrased formally in Definition 1.2, as it applies to the s-set model of solids.

8 See footnotes 3 in page 105 and 4 in page 110.

1.2 DEFINITION: Touching Assemblies

Two physical solids represented mathematically by s-sets A and B are touching if (and only if) $((A \cap B) = \emptyset) \wedge ((\bar{A} \cap \bar{B}) \neq \emptyset)$. □

Note that rather than defining an independent concept, Definition 1.2 is in fact a logical consequence of Definition 1.1 and our choice of s-sets as mathematical models for solid objects.

Open sets exhibit the kind of relationship and interaction with their environment that is desirable for modeling of the interaction and relationship of physical objects in their real-world environment. However, the boundary of an open set may be just as ill-behaved as that of a closed set. We saw in Section IV.1.2 that a definition of boundary well-behavior based on analytic functions is too restrictive. The intention of this definition was to formalize the notion of a realizable surface, rather indirectly, through the continuity and smoothness properties implied by expandability of functions into power series. Definition 1.3, below, provides a more direct definition for a realizable surface through the concept of "smooth deformations" of planar surfaces. In an earlier work [ARB82a] a similar definition was given for patchwise smooth surfaces. Definition 1.3 and the one in [ARB82a] are fundamentally the same in that they both cover surfaces of the same class of single-piece objects. The difference between the two is in part 4 where Definition 1.3 permits unions as well as intersections of open sets to construct patchwise smooth surfaces, and consequently, includes surfaces of assembled objects such as those in Figure IV.8 directly.

1.3 DEFINITION: Patchwise Smooth Surface

A Patchwise Smooth Surface is defined recursively by a finite number of applications of the following rules:

1- The empty set is a patchwise smooth surface.
2- A plane is a patchwise smooth surface.
3- The image of a plane under an arbitrary function F is a patchwise smooth surface if:
    a) F is a one-to-one mapping from E^3 onto E^3.
    b) The function F, its inverse, and all their derivatives exist (and are continuous).
4- For i = 1,2, let S_i be an open set of points in E^3 such that $\beta(S_i)$ is a patchwise smooth surface. Then, $\beta(S_1 \cup S_2)$ and $\beta(S_1 \cap S_2)$ each is a patchwise smooth surface. □

Since Definition 1.3 is recursive, the boundary of the result of any finite combination of unions and intersections of open sets with patchwise smooth surface boundaries, is itself a patchwise smooth surface. Note also, that this definition considers an aggregate of patchwise smooth surfaces as a patchwise smooth surface; thus, for example, two parallel planes are acceptable as <u>a</u> patchwise smooth surface. Therefore, whereas shapes which are not homeomorphic to a plane (e.g. disconnected surfaces, patches intersecting at sharp angles, tori, etc.) cannot be produced by smoothly deforming a plane as in part 3 of Definition 1.3, such shapes can indeed be constructed recursively in part 4 of this definition.

Part 3 of Definition 1.3 states the conditions under which a function F is considered to be a "smooth deformation". Conceptually, this definition is not far from analyticity condition of r-sets. (The function $f(x) = \sin(1/x)$ of Figure IV.2, for example, is neither

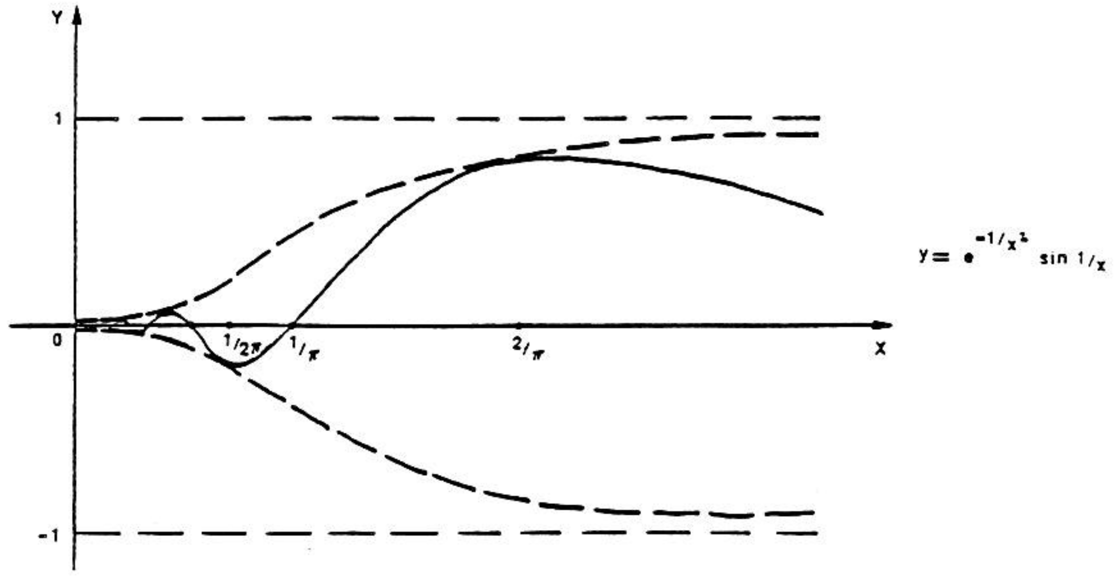


Figure IV.9 - Dampening Infinite Oscillations

continuous in the neighborhood of x=0, nor can it be made continuous by assigning any value for f(x) at x=0 [COU65].) However, analyticity prevents certain reasonable shapes, as we saw in Section IV.1.2, while our weaker conditions for smooth deformation permit a wider class of surfaces to be modeled. Unlike analyticity, smooth deformations also (intentionally) allow three-dimensional equivalents of shapes like those shown in Figure IV.9 whose "reasonability" may be considered as debatable. Such shapes exhibit an infinite number of oscillations, e.g. in the neighborhood of zero, and one may argue that they are thus unrealizable and undesirable. However, one must recognize that <u>no</u> geometric shape is in fact realizable in its ideal form. Abstract geometric shapes (e.g. a plane) can only be <u>approximated</u> to a given precision in the real world. Therefore, a shape is <u>unrealizable</u> only if its geometric properties cannot be approximated by real-world objects within reasonable precisions.

Having linked "realizability" to "a given precision", one can then argue that a surface with infinite but dampening oscillations, such as the result of extrusion of the graph in Figure IV.9, is indeed as realizable as, e.g., a plane. The reason is that any given precision imposes a bound on the number of oscillations of such a surface and makes it realizable. (Whether or not such surfaces are practically useful is a different question.) Three dimensional equivalents of geometric shapes such as those in Figures IV.6 and IV.9 are examples of patchwise smooth surfaces that cannot be defined through analytic functions. Although such shapes, as well as any other realizable shape, can indeed be approximated by analytic functions, Definition 1.3 permits direct and precise modeling of a wider class of realizable shapes than analytic functions do.

We can now give the formal definition of s-sets as:

1.4 DEFINITION: S-set

An s-set is a bounded open subset of E^3 whose boundary is a patchwise smooth surface. □

S-sets, as we have shown in this section, are suitable formal models for real solid objects. A related concept is that of three-dimensional shapes which model "almost-real solids" in the sense that, although they are legitimate shapes, they cannot be realized only because they are not bounded. We refer to these shapes as s-half-spaces since they are the counterparts of half-spaces in CSG (see Section III.2.2) and form a superset of s-sets.

1.5 DEFINITION: S-half-space

An s-half-space is an open subset of E^3 whose boundary is a patchwise smooth surface. □

Clearly, s-sets are a proper subset of s-half-spaces.

1.4 Realizable Shape Calculus (RSC)

This section proposes a new geometric solid modeling scheme, called Realizable Shape Calculus (RSC), in the same spirit as CSG, but based on s-sets. We call this scheme a calculus of realizable shapes because, as CSG, RSC expresses factual relationships between arbitrary shape-components of a part and thus, should best be regarded as descriptive, as opposed to procedural or algebraic. A twin RSC scheme, analogous to the CSG scheme based on generalized half-spaces, is also possible with s-half-spaces. The advantages of this new modeling scheme, RSC, over CSG are:

1- Unambiguous consistent representation of connected pieces
Because s-sets are open, they cannot be connected at their edges or vertices. Consequently, there is only one possible interpretation for shape configurations such as those in Figure IV.4, namely, that of Definitions 1.1 and 1.2. Furthermore, this interpretation is consistent both throughout the modeling scheme and with our real-world experience and expectations.

2-Uniform coverage and treatment of assemblies

RSC covers assemblies of objects as well as single pieces, in a uniform and consistent manner. This means that every operation defined on a piece is also formally defined on any conceivable assembly, yielding consistent results. CSG, on the other hand, is not capable of formally representing the class of touching assemblies. Accordingly, in addition to a "gluing" operator as in CSG, RSC formally defines an assembly operator.

3- Wider potential coverage of shapes

Since patchwise smooth surfaces cover a wider range of realizable surfaces than semianalytic functions do, the class of shapes that can potentially be dealt with in RSC (without resorting to approximation in the model) is broader than in CSG.

An important class of mechanical design problems involve assemblies of parts and certain of their interesting properties, e.g. balanced distribution of weight, "fixed" (i.e. "bolted") vs. "floating" (i.e. "resting") assemblies, etc. Of course, it is beyond the scope of a geometric solid modeling scheme to attempt to differentiate between such classes of assemblies. But, it is essential for such a model to be able to represent uniquely and unambiguously, all such assemblies in a manner quite distinguishable from single-piece parts. RSC is the first mathematically sound solid modeling scheme to permit a coherent and consistent treatment of assemblies.

CSG Operation	RSC Operation
$X \cup^{*} Y = \overline{\text{int}(X \cup Y)}$	$R \cup^{+} S = \text{int}\overline{(R \cup S)}$
——	$R \cup S$
$X \cap^{*} Y = \overline{\text{int}(X \cap Y)}$	$R \cap^{+} S = \text{int}\overline{(R \cap S)}$
$c^{*} X = \overline{\text{int}(X')}$	$c^{+} R = \text{int}(\overline{R'})$
$X -^{*} Y = X \cap^{*} (c^{*} Y)$	$R -^{+} S = R \cap^{+} (c^{+} S)$

Figure IV.10 - CSG and RSC Operation

Figure IV.10 summarizes the definitions of various operations in CSG (the regularized set operations) and those of their counterpart operations in RSC. Following the accepted convention, the regularized set operators are designated by a * superscript; analogously, we represent our RSC operators with a + superscript. The domain and range of CSG operations are the set of half-spaces as defined in CSG (see Section III.2.2). Likewise, the domain and range of RSC operations are the set of s-half-spaces (see Definition 1.5). Thus, in Figure IV.10, X and Y represent half-spaces whereas R and S stand for s-half-spaces. The set of all half-spaces is closed under the regularized set operations [REQ77b]. Similarly, it can be shown that the set of all s-half-spaces is closed under the RSC operations. Bounded half-spaces (i.e. r-sets) are closed under the regularized set operations in Figure

IV.10, with the exception of c^*. Analogously, bounded s-half-spaces (i.e. s-sets) are closed under all RSC operations except for c^+. In CSG $X \cup^* Y = X \cup Y$, whereas, in RSC, $\cup^+$ and $\cup$ are two distinct operations. Furthermore, we can prove that $R \cap^+ S = R \cap S$ and $c^+ R = \mathrm{int}(\overline{R'}) = \mathrm{int}(R') = \mathrm{ext}(R)$. Consequently, $R -^+ S = R \cap \mathrm{ext}(S) = R \cap (\overline{S})' = R - \overline{S}$. For s-half-spaces Q, R, and S, the following set of properties also hold:

1- $c^+ (c^+ R) = R$

2- $R = S$ if and only if $c^+ R = c^+ S$

Commutativity law:

3- $R \cup^+ S = S \cup^+ R$

Associativity law:

4- $R \cup^+ (S \cup^+ Q) = (R \cup^+ S) \cup^+ Q$

Distributivity laws:

5- $R \cap (S \cup^+ Q) = (R \cap S) \cup^+ (R \cap Q)$

6- $R \cup^+ (S \cap Q) = (R \cup^+ S) \cap (R \cup^+ Q)$

De Morgan's law:

7- $c^+ ((c^+ R) \cup^+ (c^+ S)) = R \cap S$

8- $c^+ ((c^+ R) \cap (c^+ S)) = R \cup^+ S$

Figure IV.11 depicts the effect of the CSG and RSC operations defined in Figure IV.10 on some example r-sets and s-sets. Although, as seen in Figure IV.11, the RSC operations of $\cup^+$ and $\cup$ can produce different results, this is not always the case. These operations yield different results only when objects' boundaries coincide on a "surface segment". When objects are far apart or have common vertices, edges, or "volumetric segments", $\cup^+$ and $\cup$ produce the same s-sets. Therefore, it

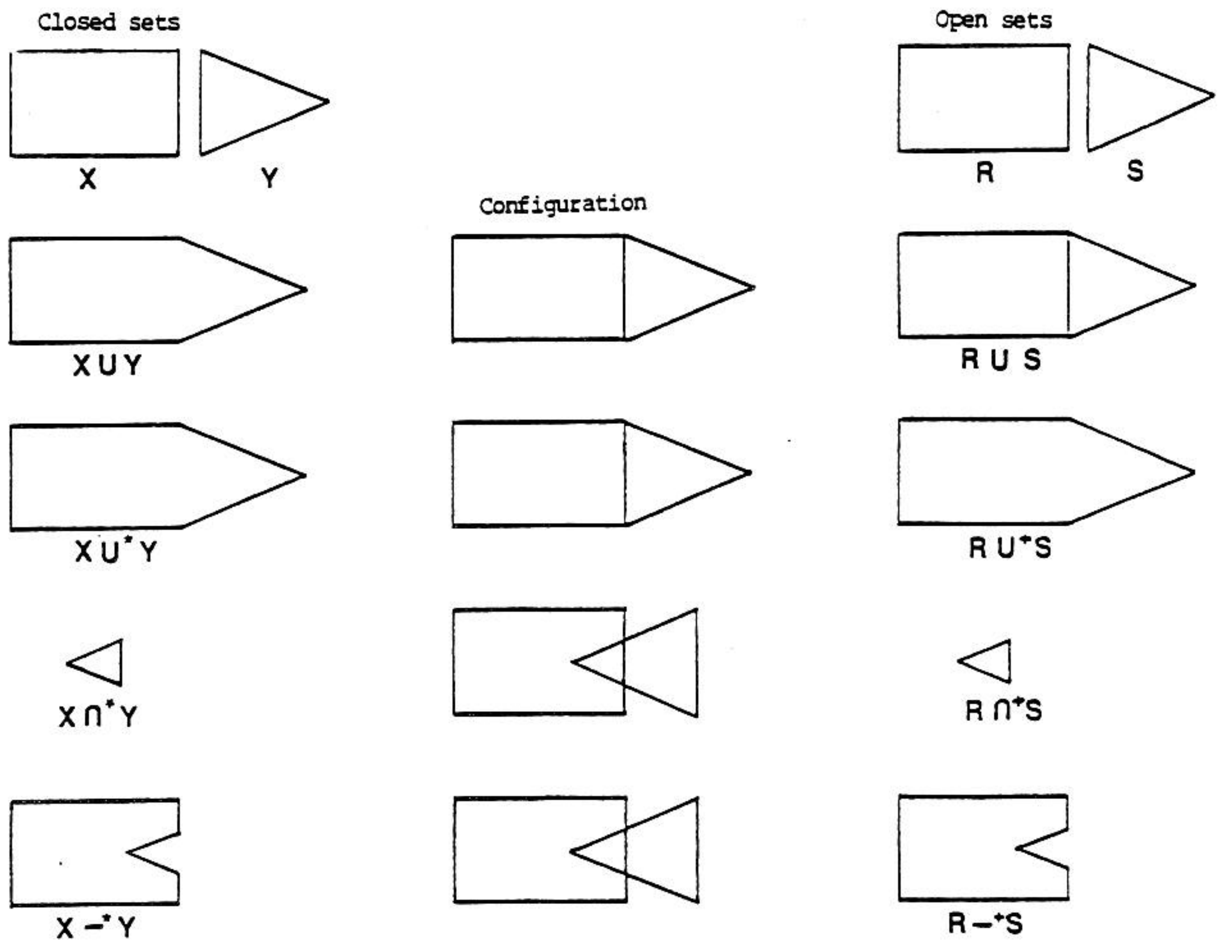


Figure IV.11 - CSG and RSC Operations on R-sets and S-sets

may be desirable to restrict the application of these operations through conditionals so that they correspond, respectively, to a "gluing" and to an assembly operation defined as:

$$R + S = \begin{cases} R \cup S & \text{if } R \cap S = \emptyset \\ \text{undefined} & \text{otherwise} \end{cases}$$

Likewise, one can restrict the application of $\cup^{+}$ on two s-half-spaces R and S, to cases where the intersection $\bar{R} \cap \bar{S}$ exists and is a three-dimensional manifold.

It is interesting to observe that the assembly operator + can, for example, be applied to the objects represented in Figures IV.12.a

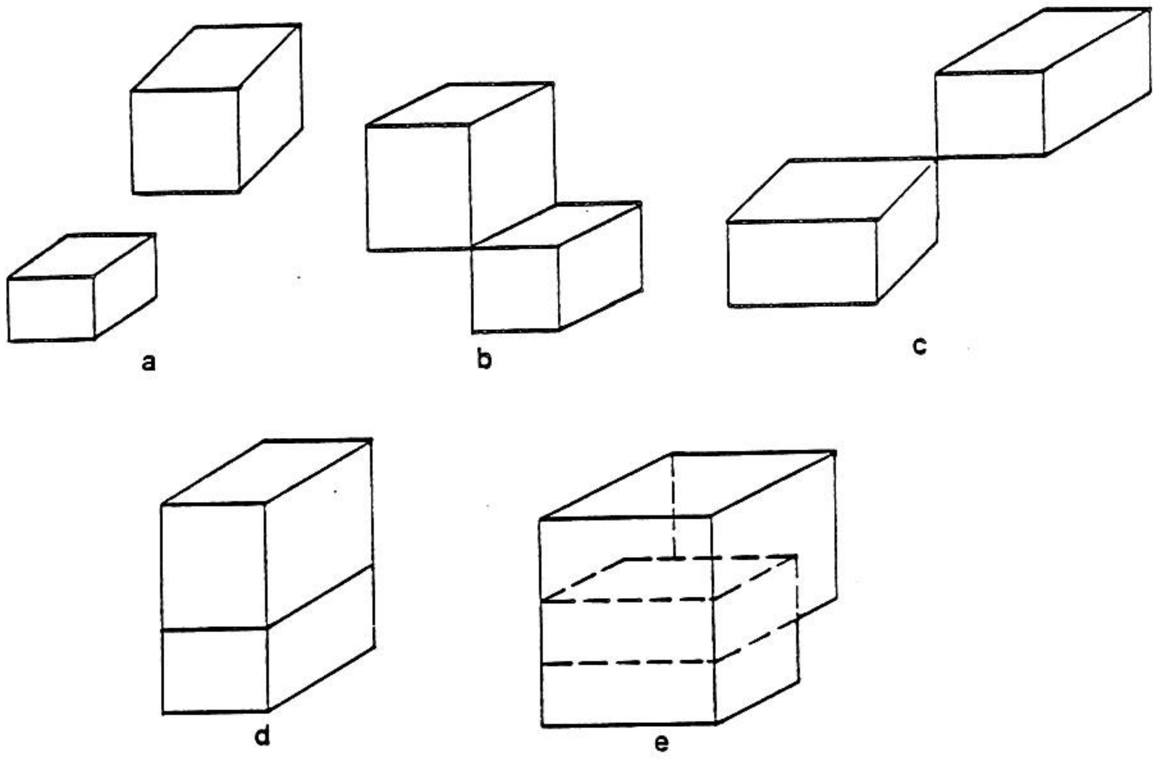


Figure IV.12 - Assembly vs. Fusion of Parts in RSC

through d, but <u>not</u> to those in Figure IV.12.e, and that the results of such applications are, consistently, two disconnected blocks in various configurations with intuitively obvious unique interpretations. The restricted version of the gluing operator U^{+}, on the other hand, is applicable only to objects like those represented in Figures IV.12.d and e and <u>not</u> to objects like those in Figures IV.12.a through c. Gluing the objects of Figures IV.12.d and e has the intuitive interpretation of "fusing" them into a single connected piece. (The result of application of the unrestricted U^{+} to the objects shown in Figures IV.12.a through c is identical to that of +.)

2 Deforming Solid Geometry

This section proposes an operational (as opposed to descriptive) approach to solid modeling based on a scheme for three-dimensional shape definition that we call Deforming Solid Geometry, or DSG. DSG is a methodology for describing mechanical parts and assemblies through a sequence of operations resembling those in manufacturing.

This methodology is based on the premise that mathematical expressiveness of a formalism alone is seldom sufficient to warrant its application to "real" problems. The reason is simple: to keep its complexity tractable, its range of expressiveness expansive, and for the sake of intellectual "neatness" and elegance, a formal deductive system is often based on a few highly-abstract axioms. This creates a gap between a formal system and the real-world problems within the context of which the deductive system was borne; a gap that, hopefully, can always in theory, be closed by rigorous deductions and systematic application of the axioms of the system. However, the burden of stretching the axioms of a formal system by deductive reasoning to reach and cover tangible entities and meaningful relationships which are entangled into a "real-life" problem, imposes a limit on the complexity of problems that can be solved effectively in that system. Thus, for example, although no modern computer is a mathematically more expressive model of computation than the simplest Turing machine, no one ever thinks seriously of solving a real computational problem on a Turing machine, and the Turing machine solution of even such relatively elementary problems as parsing, would be hopelessly intractable and incomprehensibly complex.

On the other hand, simple, elegant logical-models are useful for understanding of the nature of phenomena at play within complex systems. Compared to more life-like, less abstract models, such a model is sometimes able to provide a deeper insight by revealing more of the latent intricacies of a system.

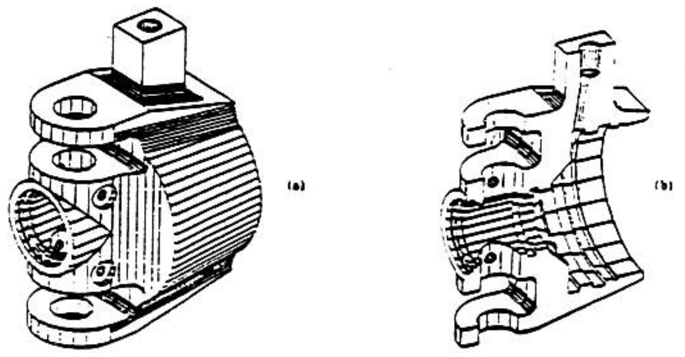
In the previous section, we proposed a new coherent formal solid modeling scheme, RSC, mathematically expressive enough to model geometric properties of any conceivable part or assembly. However, the assurance that any part or assembly can potentially be modeled in RSC, solves no real design problems, and we contend that RSC (or any CSG-like scheme, for that matter) is hardly enough ammunition for a designer of complex mechanical objects. In Section III.2.2, we concluded our review of CSG with the comments: "... CSG is a formalism for modeling of solid objects, irrespective of the way in which they can be manufactured or functionality of their components" and "... the set operators ... in a CSG tree ... [are] statements of factual relations between somewhat arbitrary shape components which do not necessarily have any functional or manufacturing-related significance or association with one another". The same critique, of course, also applies to our shape calculus, RSC, with its wider mathematical coverage. But, functional significance and associations of a part's components, as well as many manufacturing-related issues <u>are</u> (should be) the highest priority concerns of designers.

In Chapter I, we argued that an integrated CAD/CAM system is more than a CAD subsystem linked to a CAM subsystem; we argued that to develop a truly integrated system, one must start thinking in an integrated fashion, and build a framework within which, not only a designer cannot design unrealizable objects, but also, wherein he would have to sense, judge, and deal with the manufacturing-related implications of his design decisions. Such a framework should make it easier for designers to recognize and "design around" the foreseeable problems. DSG is an attempt to formulate such a framework; it can be seen as a modeling scheme, not in the same class as RSC and CSG (or boundary representation, sweeping, etc.), but which presents a better-structured and purpose-oriented view of shape composition.

An instructive and motivating analogy exists between programming languages and solid modeling schemes; they both provide a simple set of entities and operations, to be used to (creatively?) compose more sophisticated entities. The increasing impetus of programming language constructs that semantically mean what they syntactically appear to suggest, provides a supportive argument for use of shape modifying constructs in a solid modeling scheme that resemble "semantically" significant (i.e., real-world) counterparts. Modern high-level programming languages present better-structured and purpose-oriented views of computing that not only result in better-structured, purpose-oriented programs in less time with less effort, but also, tend to incite better-structured and goal-oriented <u>programming</u>. The value of such superior programming skills and methodologies goes beyond production of more elegant solutions to otherwise solvable programming problems; they permit more complex problems to be tackled.

Proliferation of CAD/CAM in the production of mechanical parts will eventually lead to automated factories; before then, however, truly integrated CAD/CAM systems must be developed. These systems will enable practitioners to grasp a better and more comprehensive picture of the activities and processes involved in fabrication of parts in general and will permit informed decisions on many technology-dependent, manufacturing-related issues early in the design phases of new parts. We contend that such decisions truly constitute as much an integral part of the design of a mechanical object as do blue-prints traditionally. We believe that, just as structured programming and disciplined programming languages and methodologies have shown to have a significant positive impact on maintainability of software (the most precious asset of computerized organizations of today), structured, disciplined design methodologies can make vital contributions to maintainability of such integrated mechanical designs, the most precious assets of automated factories of the future. Furthermore, maintainable, integrated design may prove to be the <u>only</u> sensible means for tackling design, production, and maintenance of complex mechanical systems.

Consider as an example, the rather complex part shown in Figure IV.13. This figure represents a part defined on GDP/GRIN which is used as an example in [FIT81a] to demonstrate how such complex objects can be designed using this system. Naturally, as in any other CSG-based system, there are several different ways to design this part. The particular approach taken in [FIT81a] serves their purpose well: it demonstrates the degree of flexibility and capabilities of the system. Following this approach, one quarter of the outer shape of the object is



From: [FIT81]

Figure IV.13 - A Part Defined on GDP/GRIN

first "designed", ignoring its block-shaped top. After two mirroring and a number of subtractions, the outside shape is completed as a full solid. The inside shape is "designed" separately as another solid object and is then subtracted from the first piece. Finally, the top block is added on.

The above sequence of operations certainly produce the desired shape. But, we contend, they are far from what can be considered as an integrated design of a mechanical part because they do not reflect even the slightest concern for the manufacturing-related problems which eventually must be faced for the part's fabrication. Just as software engineering contends that the goal of programming should not solely be production of programs which execute, or even those that execute correctly, we content that the goal of a design process should not solely be definition of shapes, or even definition of manufacturable shapes. An integrated manufacturable design, just like a well designed program, must also be verifiable, maintainable, and modifiable, in their broadest sense. Computers, no doubt, can tremendously aid in production of such integrated designs by providing the proper design tools in an integrated CAD/CAM environment.

The prerequisite for an integrated design approach is a structured, disciplined, manufacturing-oriented design language that combining a variety of geometric modeling schemes (e.g. RSC, sweeping, etc.) into a meaningful context, presents a consistent purpose-oriented view of design and manufacturing. DSG, we believe, is a step toward such a design language, and as such, represents a shift away from the (no doubt, necessary) line of effort in solid modeling that lead to mathematical formalizations like CSG and RSC. It is the purpose of DSG to transform the image of a design process from being conceived as a mere restatement of shape specifications of a part in terms of geometric entities, (back) into an activity through which a high-level abstract fabrication plan is delineated, the result of which is certifiably a manufacturable piece that meets those specifications. Not only this view of a design process is consistent with the drive toward factory automation and integration of CAD and CAM, we believe that a truly integrated CAD/CAM system would indeed require and impose such an approach to design. After all, automation of manufacturing, indeed eliminates (natural) intelligence and common sense from production cycles through replacement of humans with robots. As this trend continues, in order to avoid design and creation of problems whose subsequent solution would require this lacking common sense, it becomes increasingly crucial to compensate by making the one intelligent link in the cycle who is yet least likely to be replaced, the designer, more aware of the robotic behavior of other links.

2.1 Overview of DSG

In modeling schemes like RSC, geometric shape is the only primary concern and shape modification operations are intentionally "ignored" to the extent that the only potential manifestation of their existence is as (often quite non-intuitive and arbitrary) relationships between shapes, expressed in terms of set operations. (Consider, for example, expressing the relationship between the shapes of a cylindrical rod before and after a bending operation in RSC or CSG.) Shape modifying operations, thus, loose their virtues as operations in such modeling schemes. To make designers aware of the manufacturing-related implications and design (as well as manufacturing) -related consequences of their decisions, DSG incorporates abstractions of shape-modifying operations with the same emphasis as it does shapes.

Conceptually, the starting point in the design process of a part in DSG, is a sufficiently large, shapeless[9] piece of raw stock, called an "initial workpiece". This material is then "sculpted" into its final form using tools and operations that are abstractions of cutters, machine tools, and other processes used in manufacturing.

DSG consists of W, the set of all workpieces, T, the set of all tools, and a set of operations defined on W and T. In order to motivate the definitions that follow, consider a simple example where a piece of a block is to be removed by a planar cutting tool. DSG's model for this process consists of two workpieces, a tool, and an act of "application" (of the tool).

9 In the sense that its shape and size are presently of no concern.

A workpiece is an abstract data type that contains all pertinent information regarding a part, at least its geometric shape and its position and orientation in its spatial neighborhood of interest. The original block in the above example is therefore modeled by a workpiece in DSG. Likewise, the information defining the final piece after the cut constitutes the workpiece which is the result of the application. The information defining the shape of the cut (a plane in this example), its position and orientation, and which section (or side) is to be discarded after the cut, together form an abstract data type called (an instance of) a tool, in this case, a planar cutter. Informally, the "side" of a tool whose piece is to be discarded is called the void region of the tool. Apply is a function that relates (an instance of) a tool and a workpiece and yields a resulting workpiece which is obtained from the original workpiece by discarding parts of this workpiece which fall in the void region of the tool.

In addition to apply, other operations are also defined on tools and workpieces. Some operators make it possible to compose more complex tools out of simpler ones. Such composition operators can be viewed as abstractions of an extremely flexible machine tool or a manufacturing robot which can be programmed to perform complex and lengthy sequences of simple operations and manufacturing processes. Other operations, e.g. forming processes such as bending, rolling, and shaping can also be considered, as we shall see later in this chapter.

To think in terms of tools and workpieces, with its strong manufacturing orientation, is not as unnatural and foreign to designers as it may appear on the outset. Experience shows that the most popular

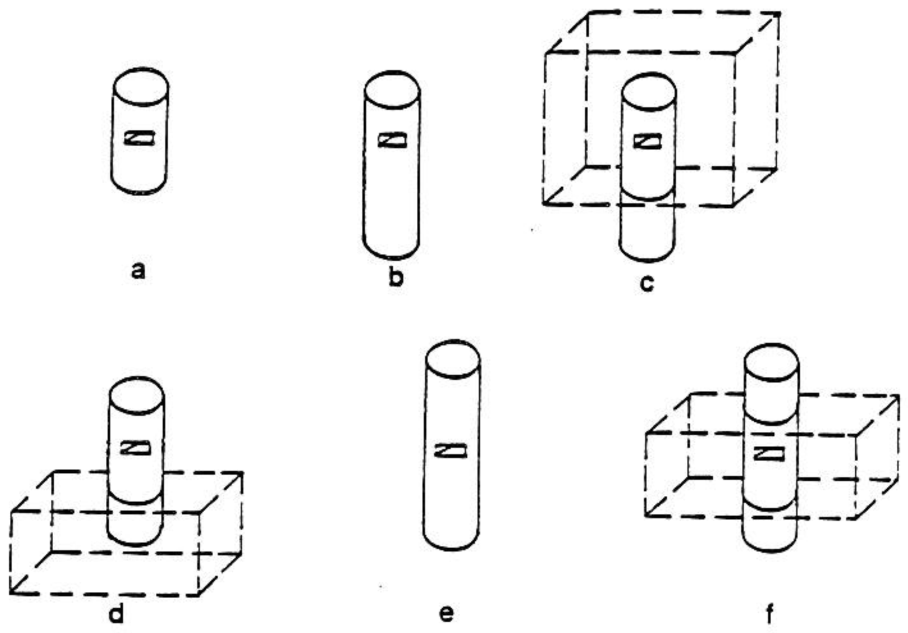


Figure IV.14 - Difference vs. Intersection

operation used in CSG-based solid modeling systems is the (regularized) difference [JOY81]. (Union, on the other hand, is not used as often, and furthermore, it is more apt to be misused as in Figure III.17 [JOY81].) The reason, we contend, is the close similarity of this operation to the material removal processes in manufacturing. We believe that preference of abstract operations with physically conceivable equivalents over directly unrealizable mathematical transformations, shows a bias in our "logical thinking" in favor of abstractions that map more directly into simple, identifiable concepts in the real world. For example, very few people would design the slotted cylinder of Figure IV.14.a given the one in Figure IV.14.b using an intersection as depicted in Figure IV.14.c; the "natural" way to get the cylinder of Figure IV.14.a from the one in Figure IV.14.b is by a difference operation as shown in Figure IV.14.d. Even when it is "less economical" to use such "natural choices", many people still do. For example, to get the cylinder of Figure IV.14.a from that shown in Figure

IV.14.e, many people would still use two difference operations to "chop off" the ends of the cylinder, rather than use a single intersection operation as in Figure IV.14.f.

Although the most successful aspect of CSG-based systems in practice is in modeling of material removal operations, CSG (or RSC) is not a very good model for such processes. In modeling of material removal operations in CSG, a problem arises from the fact that there is no distinction between what, in effect, models a material removal process (e.g. the block in Figure IV.14.d) and what represents a piece; every solid (or its complement) can become the model for a material removal process if it appears as the second operand of the regularized difference (or intersection) operator. The problem is that when used as the model of a cutting process, a solid (or its complement) is quite often an overspecification of the process which carries certain superfluous information. Such irrelevant information can cause difficulties, as discussed in Section III.2.2.

The distinction between tools and workpieces in DSG, arises from the natural distinction between manufacturing tools and workpieces in the real world. Furthermore, such distinction in our model permits clear separation of two significantly different processes: tool design and part design. The concepts of tools and workpieces also make it possible to view one of the commonly neglected and poorly handled issues in automated design, the handling of tolerances, from a perspective meaningful and natural to both designers and manufacturers. DSG views tolerances not as part definition parameters as, for example, in variational geometry [LIG82a], but rather as deviations from the desired nominal description caused by the imprecise nature of manufacturing processes, i.e. application of DSG tools and operations.

The rest of Section IV.2 develops and presents in more detail the fundamental aspects of our proposed manufacturing-oriented design language model, DSG. This discussion assumes RSC as its underlying mathematical formalism for representation of solid objects. However, it should be pointed out that reliance on RSC is merely as a convenient means for concrete presentation of ideas and the manufacturing-oriented design methodology advocated by DSG, should be judged on its own merits.

Section IV.2.2 defines the concept of a workpiece and the way it is used in DSG to represent real-world solid objects. Section IV.2.3 classifies manufacturing processes into different categories, depending on their common geometric attributes. The next two sections explore further into two such categories of operations and how they can be incorporated into a geometric model of solids. Section IV.2.6 is concerned with tolerances and how they can be meaningfully handled within the operational framework laid by DSG.

2.2 Workpieces

A workpiece in DSG is an abstract data type which defines a mechanical part or assembly, at least in terms of its geometric shape, position, and orientation. With RSC as the underlying modeling formalism, the geometric information content of a DSG workpiece is representable by an s-half-space (see Section IV.1.4). Thus, the set of all workpieces, W, includes representations for any conceivable mechanical part or assembly.

Because a workpiece is geometrically defined as an s-half-space rather than an s-set, W also includes representations which, strictly speaking, are not realizable because they are unbounded. This, of course, is intentional. The meaning that we assign to such unbounded "symbols" in our abstract data type is that they represent (bounded) real-world objects, certain dimensions of which are of no concern, provided that they are greater than a (to be determined) minimum. Thus, for example, an unbounded cylindrical shape in DSG represents a "sufficiently long" <u>bounded</u> cylinder in the real world (e.g. a rod), whose (minimum) length will be determined later. In particular, we define the <u>initial workpiece</u>, I, to be one that contains every point in E^3, with the interpretation that it represents a <u>bounded</u>, shapeless[10], sufficiently large piece of raw stock whose (minimum) dimensions (and consequently, shape) will be determined later through the operations that transform it into a desired final shape. Additionally, the workpiece that corresponds to the empty set is called the <u>null workpiece</u>. Whereas bounded workpieces correspond to s-sets that uniquely represent physically realizable objects, unbounded workpieces in DSG represent equivalence classes of realizable objects whose certain yet-to-be-determined dimensions may assume a range of values.

A consequence of allowing unbounded workpieces in DSG is the problem of deciding whether or not a given workpiece is bounded, meaning, whether or not it is least sufficiently defined as to correspond to a unique object (rather than an equivalence class of objects). Observe that a three-dimensional shape is bounded, i.e. can

---

10 In the sense that its shape and size are presently of no concern.

be fully contained in a sphere of finite radius, if and only if its orthographic projections on two intersecting projection planes are bounded (i.e. can be fully enclosed in circles of finite radii). Since there exist known algorithms for deciding whether or not a two-dimensional shape is bounded (see, for example, [KAM83]), it is always possible to determine whether a workpiece is bounded.

Checking for boundedness of a workpiece and whether or not it is null, are some of the operations defined on workpieces that inquire about certain of their attributes. The following section discusses other more interesting operations that perform various shape transformations on workpieces.

2.3 Operations on Workpieces

Having established the concept of a workpiece as the data type which represents physical objects in DSG, this section concentrates on permissible operations on this data type. For this purpose, we need to look into manufacturing processes. Manufacturing processes can roughly be classified into three categories:

1- Positioning Operations

These operations consist of translation and rotation which are characterized by the fact that they result in no shape alteration. Accordingly, the DSG version of positioning operations can be defined in terms of the three-dimensional translation/rotation operations on sets of points (see Appendix B), and can trivially be restricted such that they cannot be applied where they would result in colliding objects.

2- Assembly and Interference Checking

The characteristic of this class of operations is that they involve more than one object whose individual shapes are not modified by their application. In DSG, we model assembly by the assembly operator of RSC, + (see Section IV.1.4). Interference checking involves computation of the intersection of volumetric entities. These volumetric entities are either solid objects, or volumes swept by solid objects as they move through space. Since such volumes themselves are valid three-dimensional object shapes which thus can be represented by s-half-spaces (see Section III.2.3), interference checking in DSG essentially involves the RSC operation of intersection.

3- Shape Modifying Operations

The last class of manufacturing operations consist of those that modify the shape of a part (or assembly). We roughly subdivide these processes into two classes of material removal and forming operations. Material removal operations are an important class of manufacturing processes which due to their relative logical simplicity, are better understood in geometric modeling to the extent that they are about the only concern of the majority of CAD/CAM systems. We believe it is essential to separate material removal and forming operations into different classes, because their inherent differences prevent the same modeling approach to be successfully applied to both. The distinction between the two types of operations is that whereas in material removal processes the final shape of the modified locality of an object is rather independent of its attributes and original shape, results of forming operations heavily depend on the prior attributes of (the affected localities of) objects. In other words, material removal operations "assign" predetermined shapes to the affected locality of a part, whereas forming processes "interact" with a part to determine their results.

For example, a milling operation can be abstracted as a planar cutter. Application of such a cutter to any piece produces a planar shape (whose exact extent and dimensions, of course, depend on other case-specific parameters). Therefore, issues like roughness, plasticity, etc. can be hidden away and the planar shape of the cut can be viewed as an inherent attribute of an abstract planar cutter, meaning that it is a geometric attribute that is invariant from one application of the tool to the other. A bending operation, on the other hand, has no inherent invariant shape attributes; consider the "similarity" between the shapes of a sheet metal and a rod after they are both bent by 60°. (There are similar geometric relationships, e.g. the bend-angle, but not enough to constitute similar shapes.)

Positioning and assembly operations will not be discussed further. Sections IV.2.4 and IV.2.5 consider shape modifying operations and how they can be incorporated into our design language, DSG.

2.4 Material Removal Operations

DSG's approach to the modeling of material removal operations in principle, is not far from that in CSG or RSC; the orientation and emphasis, however, is different. Consider, for example, the simple CSG operation shown in Figure IV.14.d. There is no question that the block in this figure represents not a solid, but rather a material removal operation. However, the only useful information regarding this operation conveyed by the block, is that it leaves a planar cut surface perpendicular to the axis of the cylinder; every other information contained in the definition of the block is superfluous, irrelevant, and misleading. To do away with such superfluous information, what seems to be appropriate is to model this material removal operation by a planar half-space. But solid modeling with generalized half-spaces is even more abstract than dealing exclusively with primitive solids and by our argument in the beginning of Section IV.2, even less appropriate for a high-level design language.

The following subsections start afresh from a different view-point than half-spaces and develop an abstract model for material removal operations which will end up to be very close to RSC operations and s-half-spaces (see Section IV.2.4.2).

2.4.1 Tools

A tool is an abstraction of a sequence of possibly interleaved manufacturing processes with the net effect of a potential modification to the geometric shape of a workpiece through removal of its material. Examples of such sequences include any combination of cutting, milling, grinding, and certain types of surface finishing, possibly interleaved in time with other operation sequences. In Section IV.2.3, we argued that material removal operations are characterized by the fact that they can be associated with predetermined geometric shapes. The net effect of a material removal operation, thus, can be represented in terms of three sets defining: (1) the section of space whose material is removed, N, (2) the locus of points on the boundary created by such removal, F, and (3) the section of space whose material is left intact by the operation, P. Clearly, N, F, and P are interdependent (the most obvious relationship is that since they constitute a tri-partition of space, only two out of three need to be defined), but we temporarily delay consideration of such issues and simply define tools as tri-partitions of E^3:

2.6 DEFINITION:

A _cutting tool_ or _tool_ for short, is a triple $\langle N,F,P \rangle$ where N, F, and P are sets of points constituting a tri-partition of the three-dimensional Euclidean space. Furthermore, if $t = \langle N,F,P \rangle$ is a tool, then $V_t = N \cup F$ is called the _void region_ of tool t. □

As an example of a tool, consider a plane that divides the space into the three regions of S_l, S_r, and S, respectively designating the set of points on the opposite sides and on the plane itself. The triple $t = \langle S_l,S,S_r \rangle$ is a tool whose void region is $S_l \cup S$.

A tool is simply a specification of a material removal operation. For this specification to materialize, the operation must be performed on a workpiece. In DSG, we model this through the function _apply_:

2.7 DEFINITION: Apply

Application of a tool $t \in T$ to a workpiece $u \in W$, represented as Apply(t,u) yields another workpiece $w \in W$ which is obtained from u by removal of the part of u that falls in the void region of t; i.e., Apply(t,u) = $u - V_t$. □

2.8 DEFINITION:

Application of a tool $t \in T$ on a workpiece $u \in W$ is called proper if Apply(t,u) $\neq \emptyset$. An improper application removes the entire workpiece. Application of a tool t on a workpiece u is called useless if Apply(t,u) = u, i.e. t does not modify the workpiece. □

Figure IV.17 shows a proper non-useless application of a planar cutting tool to a block-shaped workpiece. Figure IV.17.a shows the tool intersecting with the workpiece, Figure IV.17.b shows removal of the part of the block that falls in the void region of the tool, and finally, Figure IV.17.c shows the final result.

Translation and rotation are two other operations defined on tools, with the effect of individually translating/rotating the sets of the triple. In [ARB82a], we discussed three operations called complementation, disjunctive, and conjunctive composition, for combining tools into other tools. The intention of these operations is to permit modeling of physically conceivable combinations of material removal processes. Thus, complementation reflects the fact that given a material removal process that voids the contents of a particular region of space, it is generally conceivable to devise another process that removes the contents of the complement of that region (however, there exist exceptional cases which we shall consider later). Disjunctive composition of two tools means that given any two material removal processes, it is always possible to devise another process whose net effect would be identical to successive application of the two (this third process can trivially be constructed by combining the two in a tandem). Conjunctive composition produces a tool out of two tools using complementation and disjunctive composition.

Rather than defining these operations directly in terms of set operations as in [ARB82a], we first formalize the concept of valid tools and then will define the same valid tool composition operators in terms of RSC operations.

2.4.2 Valid Tools

The intention of defining a tool as a tri-partition was to reflect the identifiable characteristics that one can associate with a material removal operation, i.e. the loci defining the shape of a cut, and the voided and intact regions of space. However, Definition 2.6 is clearly unsatisfactory because it ignores the inter-relationship of these characteristics and imposes no restrictions on the shapes that can result from application of tools. Definition 2.9, below, is a refinement that comes very close to a perfect geometric model for feasible material removal processes in manufacturing:

2.9 DEFINITION: Valid Tools

A tool $t = \langle N,F,P \rangle$ is a valid tool if the following conditions hold:

(1) P is an open set

(2) $F = \beta(P) = \beta(N)$ is a patchwise smooth surface

The set of all valid tools is represented by T_V and the two degenerate valid tools with $N = E^3$ and $P = E^3$ are denoted as t_- and t_+, respectively. □

By Definition 2.8, we can readily conclude that:

1. Application of t_+ to any workpiece is useless.

2. Application of t_- to any workpiece is improper.

3. Application of any tool to the null workpiece is both useless and improper.

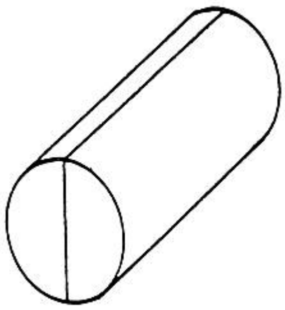


Figure IV.15 - Two Half-cylinders

Definition 2.9 states that the region of space which remains intact under a material removal operation is an open set. Since the intersection of two open sets is also open (see Appendix A), this is in accordance with our intention to model physical solids by open sets and prevents dangling faces and edges. Furthermore, this definition implies that N is also an open set. It is clear why F must be a patchwise smooth surface; no physical process can produce a three-dimensional shape which is not (a real-world approximation of) a patchwise smooth surface. The implication of requiring $F = \beta(P) = \beta(N)$ is that the cut surface in a material removal operation must become a part of the boundary of the cut piece ($F = \beta(P)$), and that, at the same time, it must be adjacent to (accessible through) the region of space whose material is being removed ($F = \beta(N)$). In other words, this condition states that a material removal process cannot create "infinitely thin cracks" such as shown in Figure IV.15 (more on this figure below). In a sense, this condition together with the requirement that our sets are open, amount to a different formalization of the useful relationship that the open-regularity condition (see Section IV.1.2) establishes between a set and its boundary.

The $F = \beta(P) = \beta(N)$ property makes it possible to uniquely define a valid tool $\langle N,F,P\rangle$ by N (or alternatively, by P). Thus, we adopt the convention of representing a valid tool by a single set. Therefore, $t=\langle N\rangle$ is a short-hand for the valid tool $t=\langle N,\beta(N),\mathrm{ext}(N)\rangle$ (note the implication that N is open and $\beta(N)$ is a patchwise smooth surface). Then, the void region of a valid tool $t=\langle N\rangle$ is $V_t = \overline{N}$.

The tool composition operators can now be directly defined on valid tools in terms of RSC operations:

2.10 DEFINITION: Complementation

The complement of the valid tool $\langle N\rangle$ is $-\langle N\rangle = \langle c^+ N\rangle = \langle \mathrm{ext}(N)\rangle$.
□

2.11 DEFINITION: Disjunctive Composition

The disjunctive composition of valid tools $\langle N_1 \rangle$ and $\langle N_2 \rangle$ is $\langle N_1 \rangle \cdot \langle N_2 \rangle = \langle N_1 \cup^+ N_2 \rangle$. $\square$

2.12 DEFINITION: Conjunctive Composition

The conjunctive composition of valid tools $\langle N_1 \rangle$ and $\langle N_2 \rangle$ is $\langle N_1 \rangle * \langle N_2 \rangle = \langle N_1 \cap N_2 \rangle$. $\square$

It is not difficult to verify that the set of valid tools is closed under complementation, disjunctive, and conjunctive composition.

We now examine the relationship between valid tools and s-half-spaces and show why s-half-spaces are too general for modeling of material removal operations. Consider the set represented in Figure IV.15. This is a valid s-set representing an assembly of two half-cylinders, and therefore it is a valid s-half-space. However, as discussed earlier, the s-set represented in Figure IV.15 is not a valid tool because it includes an "infinitely thin crack" which cannot be manufactured by any material removal operation. Intuitively, valid tools are s-half-spaces without such thin cracks and are thus a proper subset of the latter. Although shape composition using s-half-spaces is guaranteed to result in realizable shapes, using valid tools instead to model material removal processes has the advantage of requiring an explicit assembly operation in the definition of objects like the one in Figure IV.15, which in fact are assemblies, and which cannot be manufactured through material removal operations alone.

Well-structured design requires operational primitives which perform a single function each. Applying valid tools, as opposed to subtracting ($-^{+}$) s-half-spaces, prevents achieving the effect of (at least) one RSC assembly operation (+) and a removal operation ($-^{+}$) by a single removal, and enforces the requirement that important assembly operations must be defined explicitly in a design and cannot be implied.

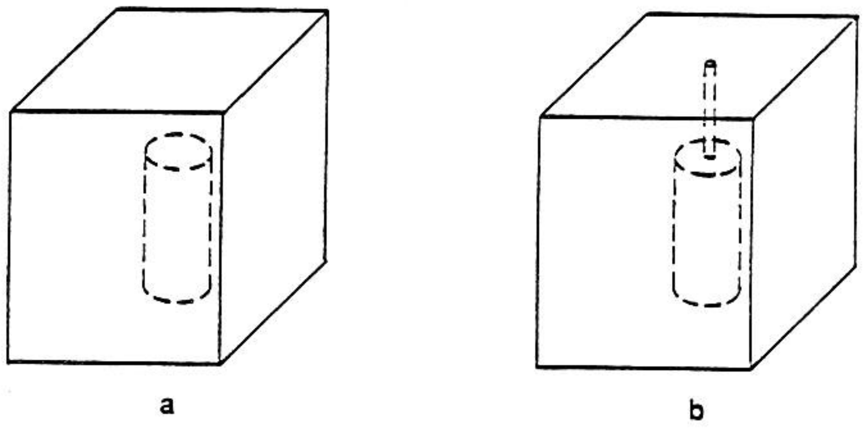


Figure IV.16 - Inaccessible Holes

One major problem still remains with using valid tools to model material removal processes. Valid tools include those which as the cylinder in Figure IV.16.a, correspond to no sequence of feasible material removal processes, involve no "assembly", and yet result in perfectly realizable shapes and valid objects. The problem with such objects is that they include holes which are not reachable from outside and thus cannot be manufactured through material removal. However, these objects are real and can be fabricated through other manufacturing processes, e.g. their two halves can be made separately which can then be fused together.

To prevent such unrealizable applications of valid tools, it is certainly possible to insist that the void region must be connected (and always remain connected to "outside" after application of tools). Imposition of this condition disallows definition of objects such as the one in Figure IV.16.a through material removal operations alone. However, even with this restriction, another class of very similar objects, e.g. the one in Figure IV.16.b, whose "unreachable holes" are connected through extremely narrow necks to the objects' environment, would still be definable through application of valid tools alone; yet these objects are equally impossible to fabricate through material removal and must be manufactured through other means such as fusion of their halves.

The above examples show that, unlike the case of the implied assembly operation, we cannot devise additional restrictions within our modeling scheme to prevent cases of "implied fusion", i.e. to prevent the RSC operator $-^{+}$ from implying a "magical" RSC gluing operation (U^{+}). This result should be neither surprising nor disappointing; a topologically oriented modeling scheme, such as RSC or CSG, cannot be expected to differentiate on the basis of purely metric criteria. What makes fabrication of the object shown in Figure IV.16.b impossible through material removal alone, is the relative size of its hole's neck as compared to a cutting device; if the neck were wide enough, it would indeed be possible to cut the hole through the neck. On the other hand, it is quite conceivable to practically avoid such undesirable part definitions in a real DSG-based system by insisting that designers should use predefined tools, and/or by relatively straightforward checking of simple metric relationships upon application of tools.

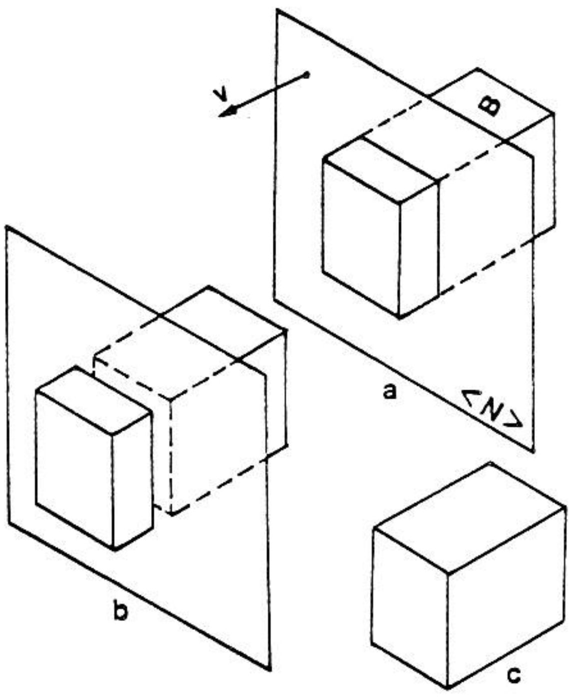


Figure IV.17 - Application of a Planar Cutter

Let us now define a valid tool generator function as any function whose range is a valid tool. As an example, observe that in E^3 a vector can be used to define a plane. Thus, if v is a vector, let P_v denote the plane to which v is normal. We can then define a tool generator function PLANE-CUT:V $\rightarrow$ T_v (where V and T_v are the set of all vectors in E^3 and the set of all valid tools, respectively) such that PLANE-CUT(v) = <N> where N = the set of points on the side of P_v that contains vector v. It is easy to show that for all v $\in$ V, PLANE-CUT(v) is in fact a valid tool.

The cutting operation of Figure IV.17 can now be expressed as a proper non-useless application of a tool <N> = PLANE-CUT(v) to the block-shaped workpiece B. The end result of this application, shown in

Figure IV.17.c is defined as apply(<N>,B) = B - $V_{<N>}$ = B - $\bar{N}$ = B $\overset{+}{-}$ N. Observe that a block can itself be created by six non-useless, proper applications of planar cutters to the initial workpiece.

A powerful tool generator function is a rotational tool generator which is an abstraction of a lathe. Let us define such a tool generator and call it LATHE. LATHE requires a planar curve defining a contour of a turned piece, an axis in the same plane as that of the curve, and a region designation for the tool, defining the curve as an outer or an inner contour. The result of this function is a valid tool described as <N>, where N is one of the two sets of points comprising the volume bounded by the surface of revolution of the contour curve around the axis; N contains the axis if the curve is designated as an inner contour, otherwise, it does not. LATHE can easily require additional properties that would guarantee conformity of the shape of the contour curve to physical limitations of available machine tools, e.g. minimum distance between the axis and curve, groove widths, step distances, etc.

2.5 Forming Operations

In Section IV.2.3 we defined forming operation as those whose shape modifying effect is dependent on the attributes of the affected piece. Generally, the most important of such attributes, e.g. roughness, temperature, pressure, etc. are non-geometric in nature. Therefore it is futile to attempt to model such operations in general, through a purely geometric scheme like RSC. However, useful special cases of forming operations which are dominantly dependent on geometric (i.e. shape related) attributes, can in fact be modeled geometrically. One such model is proposed below for an important class of manufacturing processes which we call simple bending.

Although the following discussion is aimed at development of a model for a specific class of forming processes, it addresses certain important issues that need to be considered in the modeling of other forming processes as well.

2.5.1 Simple Bending

This section presents a simplified model for a variety of manufacturing processes whose net effect can be described as a single-axis bending of a workpiece. The simplifying assumptions upon which this model is based are:

1- The non-geometric properties of a workpiece have no or negligibly marginal effect on its final shape as determined by the bending operation.

2- The workpiece maintains the shape of its axial cross-sections along the bend-length and throughout the bending process (see below).

The first assumption is necessary to permit purely-geometric modeling of any manufacturing forming process. In particular, it allows us to ignore the effect of such workpiece attributes as plasticity and temperature which could contribute to the outcome of a bending process in such a way as to, for example, determine which of the shapes in Figures IV.18.b, c, d, and e would be the result of bending the piece shown in Figure IV.18.a.

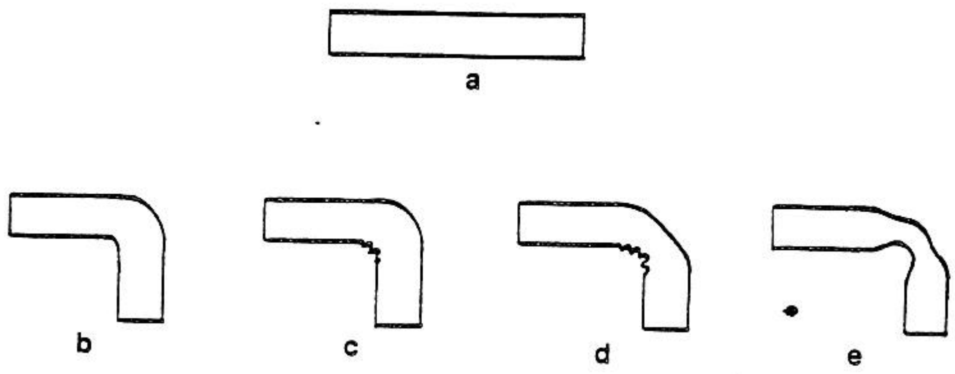


Figure IV.18 - Results of Various Bending Operations

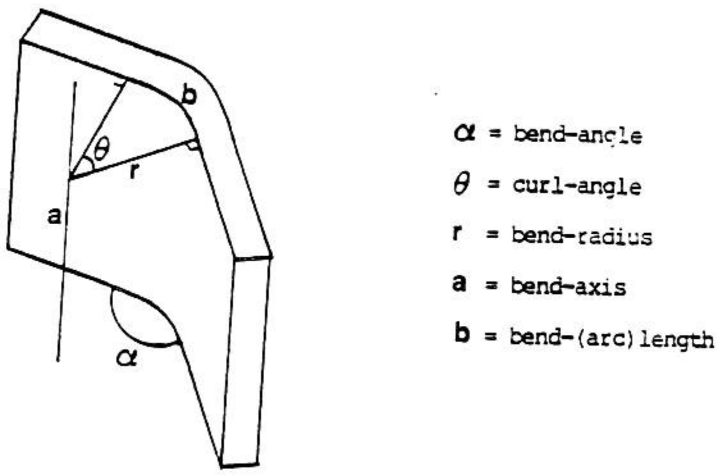


Figure IV.19 - Simple Bending

Figure IV.19 presents our terminology for bending operations. Using this terminology, axial cross-sections of a workpiece are defined on its final shape as the intersections of the workpiece with planes that pass through the bend-axis (a) and are (inclusively) contained within the curl-angle (θ). The second assumption, above, states that such axial cross-sections are all identical <u>and</u> are the same as the intersections of the workpiece along the bend-length (b), with normal planes parallel to the bend-axis prior to the bending operation. In other words, it assumes that the shape of a workpiece (i.e. its intersection with normal planes) along the bend-length is a constant before the bending, and it remains a constant after the operation, and that these two constant shapes are the same. Consequently, the geometric transformation that maps such before-bending cross-sections to their after-bending counterparts, is an identity mapping in simple bending.

We refer to manufacturing bending processes for which the above assumptions are reasonably valid as simple bending. The implication of our first assumption is that a bending operation (or indeed any class of forming operations) can be modeled by a unique set of geometric transformations, and that the parameters to these transformations are certain purely-geometric attributes. The nature (and complexity) of these transformations, however, is quite arbitrary. The second assumption, on the other hand, enables us to model a class of forming operations (simple bending) by a set of relatively simple geometric transformations. Thus, whereas the first assumption does permit us to devise a model for bending processes based on a transformation that converts the shape in Figure IV.18.a to either one of those in Figures IV.18.b, c, d, or e (it merely states that such a transformation cannot be based on non-geometric attributes), our second assumption implies that only transformation of the shape in Figure IV.18.a to that of Figure IV.18.b constitutes a simple bending.

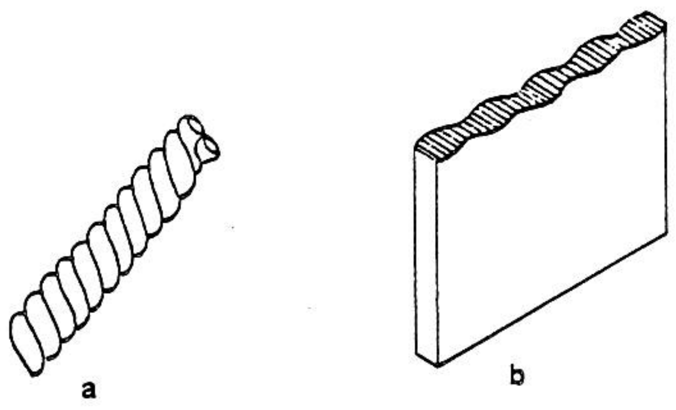


Figure IV.20 - Shapes with Varying Cross-sections

Many manufacturing forming operations such as bending of axially symmetric parts and sheet-metal bending and rolling can be reasonably well approximated by our simple bending model. Bending of certain specialized shapes, however, does not fit in this model. Figure IV.20 shows some examples. The problem with these shapes is that they defy the second assumption of simple bending: their intersections with normal planes along the bend-length are not identical to one another. It is quite conceivable to devise a uniformly-deforming transformation to maintain the shape of each such intersection through a bending operation. Thus, by somewhat relaxing the second assumption of simple bending, a more general model for bending processes can be defined to cover a wider class of shapes, including those in Figure IV.20. We do not intend to consider such alternatives further at this time because, firstly, a sufficiently large class of useful shapes and processes can be approximated by simple bending, and secondly, incorporation of a more complex geometric transformation increases the complexity of our model without resulting in significant conceptual improvements.

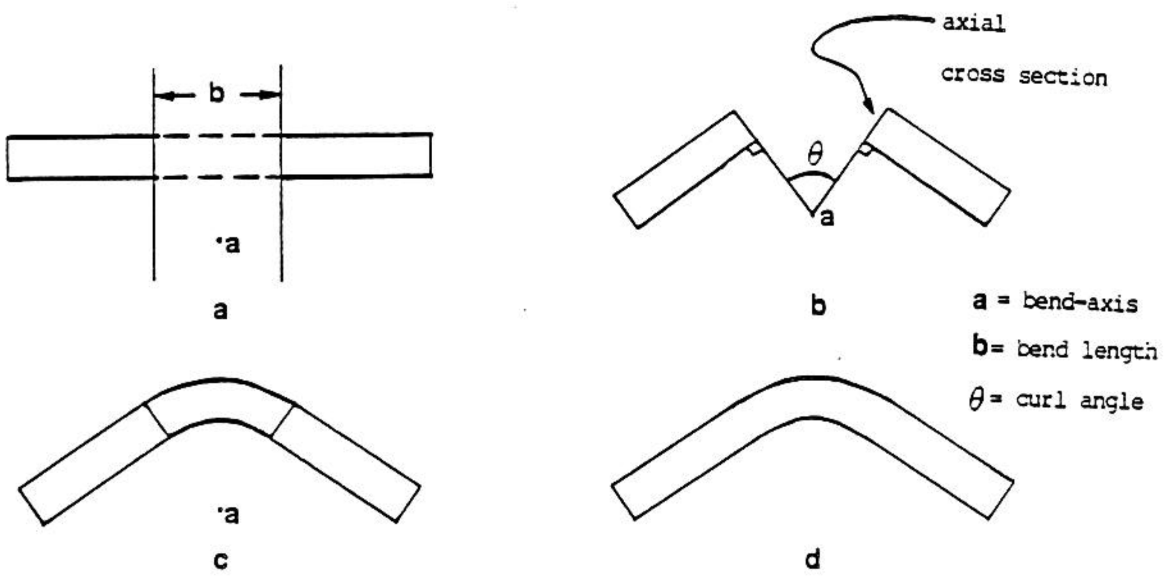


Figure IV.21 - DSG Implementation of Simple Bending

In principle, simple bending is modeled in DSG through rotational sweeping and rotation. A simple bending of a workpiece in DSG is fully defined by a bend axis and a curl-angle. All other significant parameters defined in Figure IV.19 can be derived from these two. One of the two opposite ends of the workpiece (relative to the bend-axis) can be defined as fixed for the duration of the bending operation. Consequently, one or both ends of the workpiece would be free to dislocate due to bending. The geometric transformation of simple bending can be implemented in DSG through rotational sweeping and RSC operations in four steps:

1- Bisection

The workpiece is bisected by a tool made out of two planes apart by a distance equal to the bend-length, both parallel to the bend-axis and normal to the workpiece (Figure IV.21.a). If one end of the workpiece is designated as fixed, then its corresponding bisecting plane passes through the bend-axis; otherwise the two planes are equidistance from the axis.

2- Rotation

One (or both) of the pieces of the workpiece which is (are) designated as free is (are) rotated along the bend-axis by (one half of) the curl-angle (Figure IV.21.b).

3- Sweeping

The axial cross-section of the workpiece is rotationally swept along the bend axis by the curl-angle to fill up the gap caused by bisection and rotation (Figure IV.21.c).

4- Fusion

The three touching pieces are fused together by an RSC U^+ operation (Figure IV.21.d).

Note that the above steps describe the geometric transformation that derives the final shape of a workpiece as the result of a simple bending operation. These steps or their intermediate results are not seen by either designers or manufacturers. This is an important point because it demonstrates how a meaningful high-level geometric transformation that closely models a manufacturing process can be implemented as an integral operation in DSG. In this case, DSG defines an easily understood transformation (bending) in terms of the essential parameters (bend-axis and an angle) which are directly relevant to its real-world manufacturing counterpart.

Figure IV.22 - Sharp Angle Due to Zero Bend Radius

Several variations and enhancements to this scheme for bending are, of course, possible. In particular, note that we permit a bend-radius of zero which produces a bend length of zero. This results in "sharp angles" which are not truly possible in the real-world, at least not through bending (see Figure IV.22). Exclusion of such cases and validation of the proportional dimensions of a workpiece are among the straightforward improvements that can be considered. But even in its simplest form, simple bending in DSG is a powerful operation that meaningfully combines two geometric modeling schemes (sweeping and RSC) to present an image of a useful and important class of manufacturing processes in a context easily understood by both designers and manufacturers.

2.6 Tolerance

Tolerance refers to the extent of inaccuracy that can be sustained or tolerated in a design without jeopardizing its functionality. Thus, tolerance is different than precision which is the degree of exactness to which a quantity can be measured. If an inaccuracy can be measured (precision), it is in fact always possible to devise a manufacturing process to eliminate it. Therefore, in theory, any "reasonable" precision can be met in manufacturing and there is no need for tolerancing. Nevertheless, the concept of tolerancing becomes a genuine design issue because it is the designers' job to know that all manufacturing processes are inherently inaccurate and although they can be combined in such a way as to produce accurate results to within any possible measurement precision, doing so could indeed be exorbitantly costly and absolutely unnecessary. For example, a deviation of 1/16" in a 60"x30" desk-top is by no means an obstruction to its functionality and therefore it is senseless to insist on its eradication.

In this view, it is the inaccuracies of manufacturing processes which emerge in design in the form of tolerance and (must) become a concern of designers to be judiciously weighted on the basis of their impact on functionality and fabrication costs. In contrast to geometry and shape, tolerance is much less of a "free choice" for a designer; whereas he is free to choose almost any shape or dimension to <u>realize</u> the functionality he contemplates, he is more severely limited by pragmatics and cost implications in his choice of tolerances in order to <u>preserve</u> this functionality.

In traditional hand drafting, dimensions and tolerances that appear on an engineering drawing are generally "after-thoughts". As a result, inconsistent dimensions and tolerances leading to impossible or unnecessarily expensive fabrication plans, are quite common and are sometimes caught very late (if at all) in production cycles. Since virtually all mechanical CAD systems keep and, to some extent, check the geometry of parts, it is generally difficult to produce inconsistent dimensions in Computer Aided Drafting/Design systems. On the other hand, very few CAD systems incorporate a geometric model capable of representing tolerances in general and, consequently, tolerance in these systems is still mostly an "after-thought" whose consistency and reasonability remains unverified.

The current literature on models for dimensioning and tolerancing is sparse. In PADL-1, a hierarchical tree is used to represent the set of dimensions along each of the three principal axes [REQ77a]. This technique relies on the principle of orthogonality in PADL-1 (see Section III.3.4.1) and it is not clear whether it can be successfully generalized to incorporate arbitrary geometric shapes positioned arbitrarily to compose a part. In order to model dimensions and more importantly, tolerances, it is necessary to identify an object's boundary elements (faces, vertices, and edges) upon which dimension and tolerance constraints are to be imposed. This information is not readily available in a CSG definition of a part and must be (non-trivially) evaluated [SIL81] before tolerances can be assigned to functionally meaningful dimension-chains in PADL.

Rigidity matrices of [HIL78a] and, along the same line, variational geometry [LIN81, LIG82a, LIG82b] are examples of more general mathematical models for dimensioning capable of dealing with tolerances. In [HIL78a] three types of "stiffeners" are used as constraints on linear dimensions, angle dimensions, and planarity of a set of geometric components which comprise a "flexible-joint frame" model of a solid object. These stiffeners can be combined into more complex constraints reducing a frame's degree of freedom. The rigidity matrix corresponding to such a frame indicates whether a represented object is under, over, or exactly constrained (i.e. defined) and can be used to study the geometric effect of "small loosenings" of some of its stiffeners (i.e. tolerances). Variational geometry [LIN81, LIG82a, LIG82b] incorporates a generalization of the rigidity matrices of [HIL78a] into an approach to part definition similar to that of [FIT81b]. In this approach, hand-drawn sketches are used as a rough definition of a part's "topology". Such sketches are then fitted with straight lines and arcs and the result is then used in conjunction with a set of parametric dimensions to define a solid object. Tolerance, shape, and certain functional characteristics of a defined part are represented as mathematical constraints in terms of equations binding the part's parametric dimensions. Sufficiency and consistency of such a part definition, as well as consequential geometric effects of "varying" certain parameters, can then be checked through solving a system of simultaneous equations.

Variational geometry provides a mathematical framework for study of such interesting topics in CAD/CAM as families of parts and tolerances. However, this framework alone is not sufficient for a complete treatment of tolerances because it is limited to modeling of geometric aspects of tolerancing only. On the other hand, even more so than in case of dimensioning (see Section III.1.1.2), the primary goal in tolerance specification is <u>not</u> shape definition, but rather, characteristic specification of the manufacturing operations which fabricate shapes. Variational geometry makes it possible to study the geometric effects of changing parametric dimensions of a part subject to a set of constraints. As such, this and other similar schemes, e.g. rigidity matrices of [HIL78a], form a necessary foundation for modeling of tolerances. In reality, however, not all dimensions of a part can be set or changed freely or similarly, although it may appear so mathematically. Manufacturing processes and the order in which they are applied impose a "priority" on certain dimensions and tolerances and create a "consequentiality relationship" among certain shape-components of mechanical parts. This relationship must be considered and verified in an integrated design in order to achieve a practically feasible and economically optimal fabrication plan. Such complex relationships cannot be fully represented in terms of simple constraints as those in variational geometry.

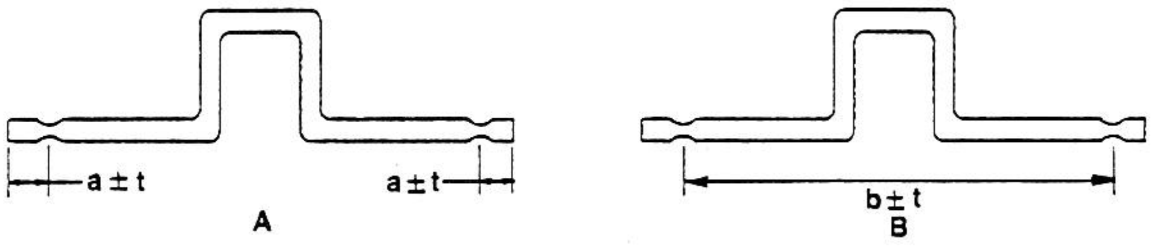


Figure IV.23 - Precedence of Operations Imposed by Tolerances

For example, consider Figure IV.23 which shows the front view of a part with two different (incomplete) dimensioning and tolerancing schemes, requiring the same <u>tight</u> tolerance t. At the outset, it may seem that the two schemes are equivalent or even, that scheme B which implies only one tight tolerance should be more economical for manufacturing. However, a closer examination reveals that not only scheme A is less costly, but that the two schemes imply two quite different manufacturing process plans, implementation of one of which (the one corresponding to scheme B), may in fact be well beyond the practical limitations of the common manufacturing tools and processes. The true nature of the problem can be appreciated only when the manufacturing process plans implied by the two schemes are considered. Both plans, naturally, involve a few machining (lathing) and bending operations. The crucial point, however, is the order in which these operations are performed. Because of the fact that the tight tolerance involved here cannot be met through a bending operation, scheme B implies that the rod must be bent before its final machining, whereas scheme A permits a manufacturer to perform the lathing operations first and then bend the rod. A very serious problem now unveils: depending on the actual dimensions of the part, it may in fact be impossible to lathe a bent piece as per scheme B. Moreover, when and if possible, such an operation is very likely to be much more expensive than a regular lathing process such as what is required for scheme A. The fundamental issue here, is the presence of a tight tolerance in scheme B, over a length whose fabrication involves an inherently imprecise process, i.e. bending. Scheme A, on the other hand, places the same tight tolerance over lengths whose fabrication involves two more-precise operations of lathing and end-facing (cutting). It is not possible to even begin to appreciate and deal with the variety of subtle points such as those alluded to in the above example, with tolerance modeling schemes like variational geometry or the ones that can be incorporated into a purely shape-oriented model such as CSG [REQ77a] or RSC, where the fundamental protagonists (i.e. manufacturing operations) are not recognized, represented, and treated properly. Techniques such as rigidity matrices [HIL78a, b and c] and variational geometry, however, constitute a suitable mathematical basis for the study of the geometric effects of dimensioning and tolerancing.

We believe that a proper model for dimensioning and tolerancing should be based on an operational cause-effect view of part production: (positional) dimensions define the nominal locations affected by manufacturing processes, whereas tolerances are caused and imposed on a design by the manufacturing processes selected for its fabrication. Since it is the responsibility of designers to ascertain that a part's functionality is most economically preserved by tolerances which result from manufacturing operations, designers' contribution to manufacturing process planning must go beyond "conservative" specification of a set of "idealized" tolerances. Designers must verify tolerances, not only to confirm their consistency and accordance with a part's functionality, but also to validate their economical justifiability in view of their implications on the part's manufacturing process plan.

The operational framework of DSG presents an appropriate foundation for modeling of tolerances not only in terms of their mutual consistency and effect on geometry, but more importantly, in terms of their impact on fabrication cost and process planning. Detailed development of a sound formal model for tolerancing is a recognized necessity which is beyond the scope of this dissertation. Certain important aspects of tolerancing, however, can be successfully handled through DSG by associating a tolerance with every defined tool and operation. Application of tools and operations in the course of definition or modification of shapes, would then both define consistent dimensioning schemes, and impose their associated tolerances on the affected shapes. In this way, such issues as feasibility, consistency, and cost can be automatically handled in a DSG-based system through safeguards and validity checks initiated by designers' tool selections.

3 Examples of Part Definitions in DSG

The following examples are intended to illustrate how a DSG-based system would be used by a designer in order to define a mechanical part.

As a first example, consider definition of a piece of raw stock shaped as a hexagonal rod. Such a piece can conceptually be "carved" out of the initial workpiece by application of six planar cutters at 60^{o} angles to each other, followed by two other planar cuts perpendicular to the first six in order to end-face the stock. The information required for positioning and application of these cutters can be entered interactively through a graphic interface into a DSG modeler. A designer can, for example, draw a two-dimensional hexagon and subsequently use it as a means of supplying the information that defines the position and orientation of the first six planar cutters[11] (see Figure IV.24.a).

A design scenario could be as follows: The designer first selects the initial workpiece (or equivalently, a sufficiently large block) from the system's raw stock menu. By selecting a planar cutter with proper tolerance (see Section IV.2.6) from the menu of tools, he then informs the system what tool he intends to apply. The user-interface processor now knows the specific modeler function to be invoked and the

11 Note that this hexagon is and never will be a part of the object's model; it is merely a "graphic expression" which gets evaluated by the interactive graphic user-interface into the necessary parameters expected by the DSG modeler. In order to avoid confusion, such graphic expressions would better be represented in a distinct color, if possible, or a specific line-type. We use dashed-lines to show any graphic symbol which is not a part of solid models.

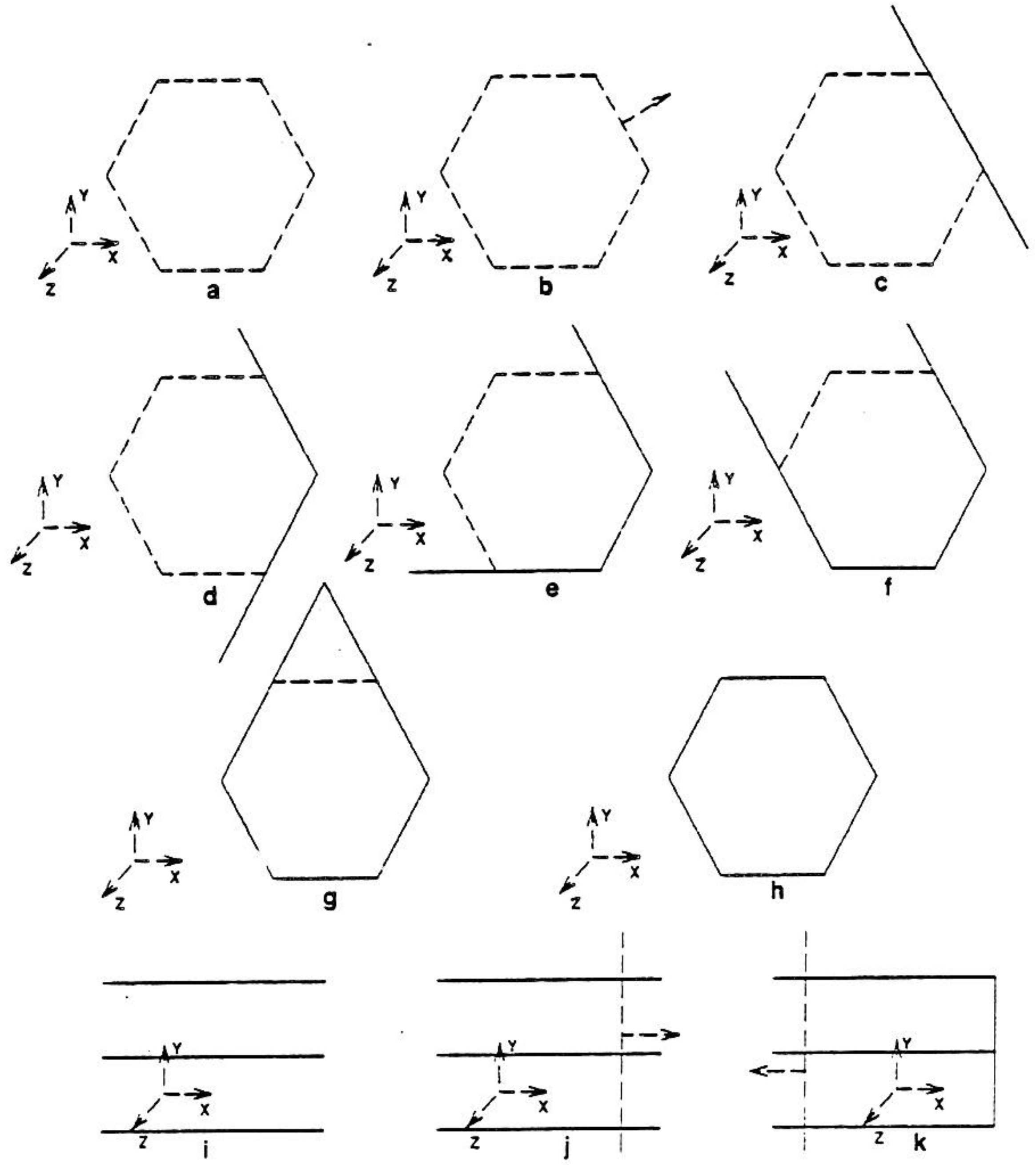


Figure IV.24 - Definition of a Hexagonal Rod

information it requires as parameters. In our example, this function is PLANE-CUT defined in Section IV.2.4.2 and expects a vector as its parameter. Selection of an edge of the two-dimensional hexagon together with a "direction" indication by the designer, provide the user-interface processor with sufficient information to create a suitable vector (Figure IV.24.b). Supplied with this vector, PLANE-CUT generates a valid tool (see Section IV.2.4.2) whose application to the initial workpiece results in another workpiece which is bounded only on one side (see Figure IV.24.c). Similar applications of PLANE-CUT using other edges of the two-dimensional hexagon yields the workpieces shown in Figures IV.24.d through h. Meanwhile, the (partially completed) workpiece can be moved and rotated at will, in order to provide the designer with his desired view. A 90° rotation of the (still unbounded) rod in Figure IV.24.h around the y-axis positions the rod properly for end-facing (Figure IV.24.i). Two more applications of PLANE-CUT, a proper distance apart (Figures IV.24.j and k), result in the object shown in Figure IV.25.

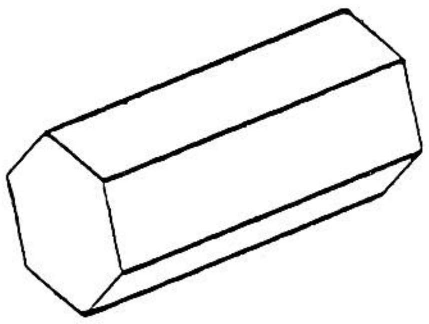


Figure IV.25 - Hexagonal Rod

Since application of each tool modifies the geometric shape of the workpiece, assigns a tolerance to its modified locality, and at the same time, specifies a sequence of (interleaved) manufacturing operations which can materialize such desired effects, the final model is that of a consistently-toleranced solid object which is well-defined both geometrically and in terms of a feasible, abstract fabrication plan. At this point, definition of the hexagonal rod can be stored into the system's raw stock menu (possibly, together with a set of constraints relating characteristic dimensions of its shape for validation checking upon use), in which case it can be automatically verified to insure that it indeed defines a finite piece.

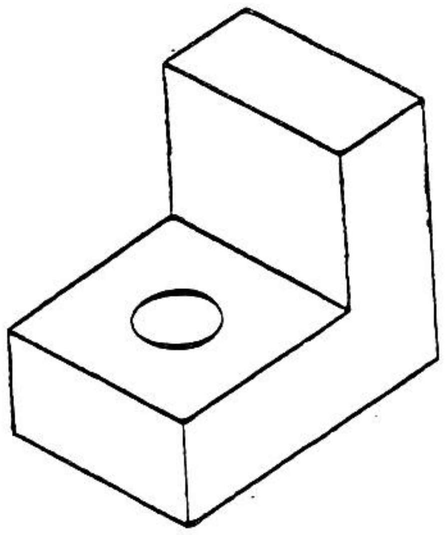


Figure IV.26 - L-Shaped Bracket

As another example, consider definition of an L-shaped bracket shown in Figure IV.26. A scenario for this definition could start with selection of a sufficiently large piece of block-shaped raw stock from the menu. Several applications of planar cutters produce an exact sized block with desired tolerances. Next, an L-shaped corner-cutter (conjunctive composition of two perpendicular planar cutters) with proper tolerance produces the bracket. Finally, the hole can be "drilled" by application of a cylindrical cutter with proper diameter and tolerance.

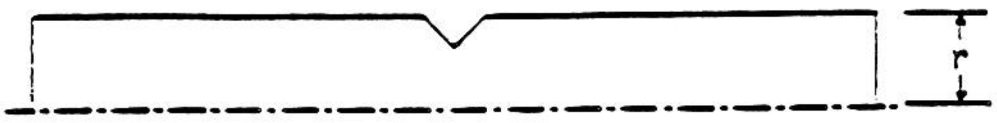


Figure IV.27 - A Lathing Contour

As a third example, consider a simple lathing operation on a part such as the one in Figure IV.25. The lathing contour and axis for this operation can be defined through the interactive graphic interface as in Figure IV.27. This information, together with a tolerance and another parameter indicating that this is an outer contour, is then passed to function LATHE (see Section IV.2.4.2). LATHE in turn, can verify that the requested operation and tolerance are indeed feasible before it generates a valid tool. Application of this tool to the workpiece shown in Figure IV.25 results in the object shown in Figure IV.28.

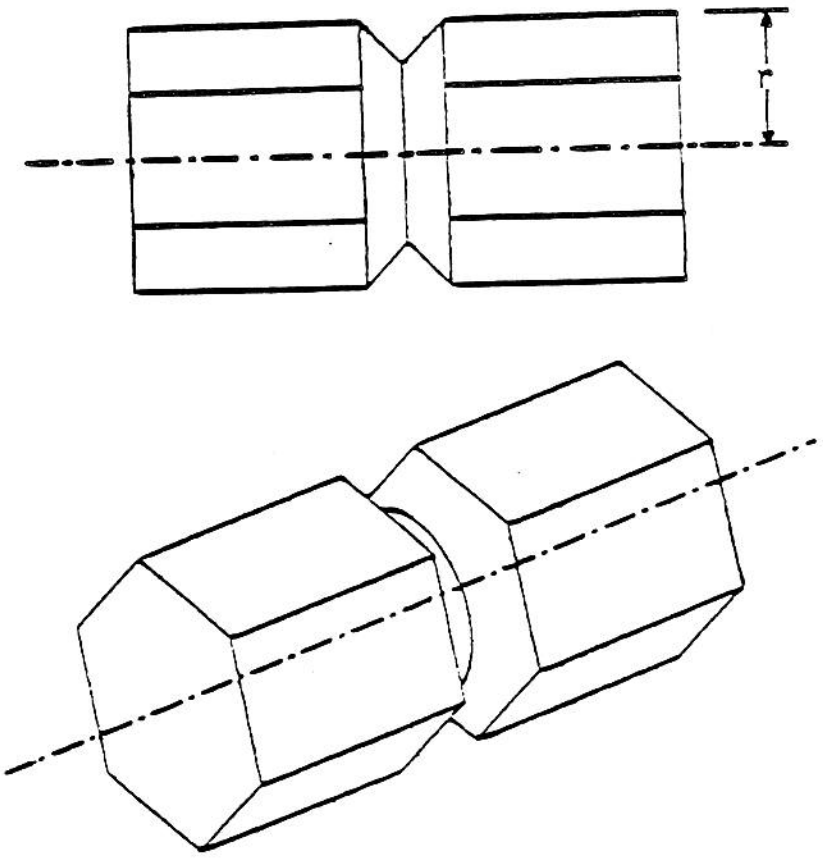


Figure IV.28 - Two Views of a Turned Piece

4 Conclusions

DSG proposes use of abstractions of manufacturing tools and processes during the design of mechanical parts; it does not advocate, however, that a design process should define or even simulate a part's actual manufacture. In other words, the DSG model of part definition falls somewhere between a simulation model of its manufacturing, and on the other extreme, the highly abstract shape composition model of RSC.

In general, every abstraction eliminates certain information from the model it produces. Depending on the degree of abstraction, however, such eliminated information can be considered "hidden" or "lost". This distinction is important in modeling because although neither lost nor hidden information is contained in a model itself, hidden information can generally be seen and manipulated indirectly through implication, whereas lost information cannot be accessed through the model. For example, the details and the precise order in which real manufacturing operations are performed to fabricate a part are eliminated from both RSC and DSG models of the part. However, this information is hidden in DSG in the sense that the existence of a set of shape modifying operations subject to real manufacturing constraints is acknowledged and, in fact, constitutes an integral part of this model. This same information is lost in RSC (or CSG) in the sense that, in general, a part definition in RSC reflects no hints to or concerns for real manufacturing constraints. Consequently, DSG's procedural style of part definition and similarities between DSG operations and those in actual manufacturing should provide a more suitable basis for automated process planning than CSG or RSC definitions do.

CAPP codes (see Chapter I) in effect define equivalence classes of parts based on certain characteristic attributes of their geometric shapes. These characteristic attributes are defined in such a way as to determine a certain sequence of manufacturing operations each. A part definition in CSG provides no relevant information other than the part's geometric shape. Association of CAPP codes to such a purely-geometric definition, in general, requires some quite non-trivial heuristics. On the other hand, DSG defines equivalence classes on (sequences of) manufacturing processes and operations, based on the shapes and tolerances they produce. Each such equivalence class is represented in DSG by an abstract tool or operation. A carefully defined set of tools in a DSG-based system can in fact come quite close to defining the same equivalence classes of shapes as does a specific CAPP-coding scheme. Conceptually, in such a system, the DSG definition of a part can by itself substantially determine its CAPP code. Thus such DSG definitions become an aid to automated manufacture of parts by providing the same geometric information as contained in an equivalent CSG definition, in a more-directly-usable format for CAM.

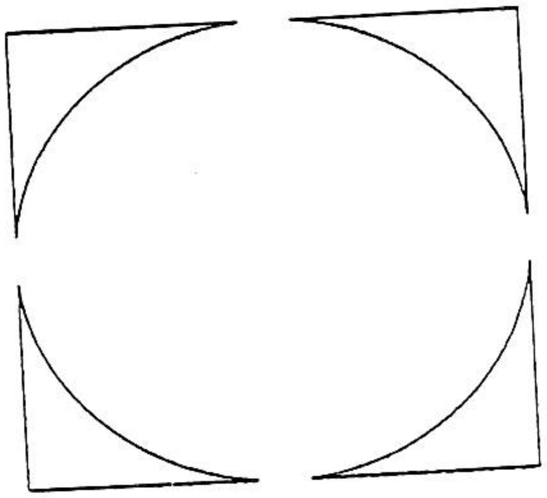


Figure IV.29 - An Assembly of Wedges

Given a DSG part definition, process planning involves selection of a particular (sequence of) manufacturing operation(s) among the members of the equivalence class of operations associated with each DSG tool used in that definition. This selection is based on a number of parameters, some of which can be geometric in nature. A judiciously defined set of tools in a DSG-based system should minimize the number and complexity of such geometric parameters. For example, the assembly of Figure IV.29 can be defined in DSG by application of a cylindrical cutter drilling a hole in a block. This definition, however, does not correspond to a feasible manufacturing process plan and therefore at process planning, instead of a drilling operation, another member of the equivalence class associated with this cylindrical cutter must be "selected", i.e. one that corresponds to a more elaborate sequence of material removal operations which together produce the assembly. The criteria for selection of this sequence of operations over a simple drilling is geometric in nature, and automatic derivation of these operations can be complex. A DSG-based system can choose not to allow such definitions in the first place, by imposing validity checks on tools upon their application (e.g. the hole's diameter must be smaller than the size of the block in Figure IV.29).

The choice of permitting a designer to use a cylindrical cutter to define the assembly of Figure IV.29, vs. forcing him into defining this assembly in terms of more elaborate, but realistic manufacturing operations, typifies the two ends of the spectrum of abstraction wherein DSG part definitions lie. It is conceivable to develop a DSG-based system flexible enough to permit a wide range of abstraction in part definitions. In such a system, early stages of a design can use more-abstract, simpler operations to initially define a shape (e.g. a cylindrical cutter to produce the assembly of Figure IV.29). Later (maybe at process planning), such unrealistic operations can be (semi)automatically located within a definition and substituted with (i.e. "compiled" into) more elaborate sequences of operations.

A potential disadvantage in using DSG as a design language is that the operational structure and discipline imposed by DSG on designers can restrict their creativity. The seriousness of this point, of course, remains to be evaluated in practice. However, the need for standardization in design and for a structured and disciplined design methodology has been acknowledged by many researchers, e.g. [GJO80, SAM76], and one can argue that any discipline is a restriction on creativity. Furthermore, no design can in fact be considered complete unless at least the (sometimes elaborate) details needed for its DSG definition are worked out. Thus, the extra effort (if any) required for completion of a DSG part definition (as, for example, compared to an RSC definition) is in fact an effort spent on completion of an integrated design, rather than to overcome some artificially created barrier.

There is more to the issue of creativity in design than our brief discussion above. An overwhelming majority of real-world designs can indeed be categorized as "routine design". As opposed to "creative design", routine design involves a part whose functionality is quite clearly understood, its shape is almost precisely defined, its manufacturability is guaranteed, its fabrication plan is quite well understood, and for which a very accurate production cost estimate is also available. Examples of this type of design include design of such common items as bolts, sleeve stops, support brackets, etc. Virtually all new part designs which are in fact obtained through a modification to an existing object, also fall into this category. The significant characteristic of routine design is that it does not involve creation of new tools or manufacturing processes. On the other hand, creative design involves creation of truly new parts in order to materialize new functionalities. Fabrication of such parts may require new and specialized tools and processes, the precise design of which should carefully go hand-in-hand with that of the parts themselves.

We contend that DSG is indeed a proper framework for routine design of mechanical parts. It is only in creative design that simultaneous involvement with functionality, geometric shape, and manufacturing issues may confine designers' creativity. Perhaps what is required for creative design, then, is availability of two concurrent, compatible, but different systems: a DSG-based system on which complete detailed designs can finally be produced, and another system (RSC-based? surface oriented? based on free-hand sketching? ...) on which preliminary design "sketches" can be tried out free of manufacturing concerns, until a reasonably firm geometric shape evolves for a new part.

In Chapter III we reviewed several major representation and modeling schemes for solid objects. In Section IV.1.4, yet another solid modeling scheme was proposed. Aside from formal limitations of some of these schemes, many of them are indeed practically useful models, each with a narrow but focused view of geometric shape composition which makes it suitable for certain specific design purposes. Amalgamating several geometric modeling schemes in a CAD system, theoretically increases the system's potential applicability and expressiveness. But, diversity of incompatible views in such a system can in fact become an obstacle to its effective usefulness. To provide a meaningful multi-approach solid modeling scheme, a broader coherent view of object definition is required which is beyond that of any single specialized approach to solid modeling such as RSC, sweeping, boundary representation, etc.; what is needed is a context wherein a number of these solid modeling schemes can "naturally" fit. We believe that a design language based on a manufacturing-oriented view of part definition provides a broad enough coherent context within which a variety of different geometric modeling approaches can concur.

The impetus of integration of CAD and CAM is a confirmation of the fact that the artificial isolation of design and manufacturing is no longer productive in an overall analysis. This integration, however, cannot be meaningfully achieved unless a manufacturing-oriented context is laid as the foundation of a design language wherein fabrication limitations and economics of design can be defined and dealt with by designers. Reflecting the operational nature of manufacturing in design permits controlled and selective exposure of designers to various hierarchical levels of such details in an integrated design of mechanical parts. An operational approach to design also makes it possible to coherently deal with issues like practical applicability of tools and operations, tolerances, and forging envelopes, through associating certain "semantic attributes" with every tool and operation.

Section IV.2.6 discusses how tolerances can be modeled as semantic attributes of DSG operations. The problem of computing casting or forging envelopes for parts, although practically very important, has not been studied seriously and is virtually ignored by all CAD systems. This is, at least partly, due to the fact that the common purely-geometric solid models cannot be expected to successfully differentiate between geometric shape-components on the basis of essentially non-geometric criteria. The problem of forging envelopes can be stated as follows: Given the desired shape of a mechanical part (after machining) together with a set of required (non-geometric) properties such as strength, roughness, etc., what is the shape of an initial piece which can be machined into that desired final shape, subject to the (geometric and non-geometric) constraints imposed by the required machining operations and the part's shape?

A manufacturing-oriented operational model such as DSG makes it possible to express this problem in terms of concepts defined within the modeling scheme. The forging envelope problem can be restated in DSG as follows: Given a shape S with an associated set of attributes A_S, and given a DSG operation (sequence) O whose immediate result is S, what is the "minimum" (i.e. weakest) set of requirements that must be satisfied by a shape S' on which O can operate to produce S? In other words, find S' (and $A_{S'}$) such that application of O on S' results in S (and A_S). Stated in these terms, the similarity between this problem and correctness proofs of programs based on Hoare's axiomatic semantics of programming languages is compelling. It is both interesting and theoretically possible to define "semantics" for manufacturing operations in DSG through axioms that relate shapes and other non-geometric attributes of their operands to those of their results. A DSG part definition not only defines a part's geometry, but also gives the DSG operations which produce that geometry. Evaluation of a part's forging envelope, then, can be accomplished constructively through a sequence of formal deductions using the axiomatic semantics of its defining DSG operations, beginning with the final part and going backward one operation at a time, in the same style as a program's weakest precondition is evaluated using axiomatic semantics of programming languages.

The direction of our future work in CAD/CAM is influenced by our belief that meaningful integration of CAD and CAM requires deeper understanding of manufacturing production cycles and, in particular, further study of manufacturing processes from a design perspective. We intend to pursue this research along the following lines:

1- Implementation of a prototype DSG-based integrated CAD/CAM system

2- Automated derivation of manufacturing process plans from DSG part definitions

3- Study and modeling of tolerances

4- Study and modeling of shaping, forging, casting, and other interesting manufacturing processes

APPENDIX A

Properties of Metric Spaces

The distance between two points $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$ in the n-dimensional Euclidean space E^n is defined by the function $\delta(x,y) = \sqrt{(y_1-x_1)^2+\ldots.+(y_n-x_n)^2}$. The function $\delta(x,y)$ is non-negative for every pair (x,y) and has the following properties for any three points x, y, and z:

i - $\delta(x,y) = 0 \Leftrightarrow x = y$

ii - $\delta(x,y) = \delta(y,x)$

iii - $\delta(x,z) \leq \delta(x,y) + \delta(y,z)$

Abstracting the notion of distance from Euclidean spaces, a _metric space_ is defined as a set X together with a non-negative function δ on X x X satisfying the above three conditions [EDM80]. In such a metric space, function δ is called the _metric_ and members of X are called points. We are not interested in metric spaces in general, rather, we wish to use some of the fundamental definitions and theorems of metric spaces as they apply to Euclidean spaces, or more specifically as they apply to E^3.

A.1 DEFINITION:

Let X be a metric space with metric δ. If x_0 is a point of X and r is a positive real number, the _open sphere_ sphere(x_0,r) with center x_0 and radius r, is the subset of X defined by

$$\text{sphere}(x_0,r) = \{x \mid \delta(x,x_0)<r\}$$

An open sphere is always non-empty, because it at least contains its center. Similarly, the <u>closed sphere</u> sphere$[x_0,r]$ with center x_0 and radius r, is the subset of X defined by

$$\text{sphere}[x_0,r] = \{x \mid \delta(x,x_0) \leq r\}$$

A closed sphere too, is always non-empty, because it at least contains its center. □

A.2 DEFINITION:

Let X be a metric space. A subset G of X is called an <u>open set</u> if, given any point $x \in G$, there exists a positive real number r such that sphere$(x,r) \subseteq G$. □

A.1 Theorem:

In any metric space X, the empty set $\emptyset$ and the full space X are open sets [SIM63]. □

A.2 Theorem:

In any metric space X, each open sphere is an open set [SIM63]. □

A.3 Theorem:

Let X be a metric space. A subset G of X is open if and only if it is a union of open spheres [SIM63]. □

A.4 Theorem:

Let X be a metric space. Then (1) any union of open sets in X is open; and (2) any finite intersection of open sets in X is open [SIM63]. □

A.3 DEFINITION:

Let X be a metric space, and let A be a subset of X. A point x in X is called a limit point of A if each open sphere centered on x contains at least one point of A different from x. □

A.4 DEFINITION:

Let X be a metric space. A subset F of X is called a closed set if it contains each of its limit points. □

A.5 Theorem:

In any metric space X, the empty set ∅ and the full space X are closed sets [SIM63]. □

A.6 Theorem:

Let X be a metric space. A subset F of X is closed if and only if its complement F' = X - F is open [SIM63]. □

A.7 Theorem:

Let X be a metric space. Then (1) any intersection of closed sets in X is closed; and (2) any finite union of closed sets in X is closed [SIM63]. □

A.5 DEFINITION:

Let X be a metric space, and let A be a subset of X. The closure of A, denoted by $\overline{A}$, is the union of A and the set of all its limit point. □

A.8 Theorem:

Let X be a metric space, and let A and B be subsets of X. Then [SIM63, REQ78]:

(1) $\overline{A}$ is the smallest closed superset of A, i.e. every closed superset of A contains $\overline{A}$.

(2) A is closed if an only if $A = \overline{A}$.

(3) $\overline{A}$ is equal to the intersection of all closed supersets of A.

(4) if $A \subseteq B$ then $\overline{A} \subseteq \overline{B}$.

(5) $\overline{A \cup B} = \overline{A} \cup \overline{B}$

(6) $\overline{A \cap B} \subseteq \overline{A} \cap \overline{B}$

□

A.6 DEFINITION:

Let X be a metric space and A a subset of X. A point x in X is called a <u>boundary point</u> of A if each open sphere centered on x intersects both A and its complement $A' = X - A$. The <u>boundary</u> of A, represented as $\beta(A)$, is the set of all boundary points of A. □

A.9 Theorem:

Let X be a metric space and A and B subsets of X. Then [SIM63, REQ78]:

(1) $\beta(A) = \beta(A')$

(2) $\beta(A) = \overline{A} \cap \overline{A'}$

(3) $\beta(A)$ is a closed set

(4) A is closed if and only if $\beta(A) \subseteq A$

(5) $\overline{A} = A \cup \beta(A)$

(6) $\beta(A \cup B) \subseteq \beta(A) \cup \beta(B)$

(7) $\beta(A \cap B) \subseteq \beta(A) \cup \beta(B)$

(8) $\beta(A - B) \subseteq \beta(A) \cup \beta(B)$

□

A.7 DEFINITION:

Let X be a metric space and let A be a subset of X. A point in A is called an <u>interior point</u> of A if it is the center of some open sphere contained in A. The <u>interior</u> of A, denoted by int(A), is the set of all its interior points. The <u>exterior</u> of A, denoted as ext(A), is the complement of $\bar{A}$. □

A.10 Theorem:

Let X be a metric space, and let A and B be subsets of X. Then [REQ78]:

(1) A = int(A) if and only if A is open.

(2) if $A \subseteq B$ then $\text{int}(A) \subseteq \text{int}(B)$.

(3) ext(A) = int(A')

(4) $\text{int}(A)' = \overline{A'}$

(5) $\text{int}(A \cap B) = \text{int}(A) \cap \text{int}(B)$

(6) $\text{int}(A) \cup \text{int}(B) \subseteq \text{int}(A \cup B)$

(7) $\text{int}(A) \subseteq \text{int}(\bar{A})$

(8) $\text{int}(A - B) = \text{int}(A) - \bar{B}$

(9) $\bar{A} = \text{int}(A) \cup \beta(A)$

□

APPENDIX B

Translation and Rotation

B.1 DEFINITION:

Translate:RealxRealxRealxS → S, where S is the set of all subsets of E^3, returns Translate(a,b,c,u) = v where:

$$v = \{q \mid p \in u \text{ and } X_q = X_p + a,\ Y_q = Y_p + b,\ Z_q = Z_p + c\}$$

and X, Y, and Z represent the coordinate values of their subscript points. □

B.2 DEFINITION:

Rotate:RealxRealxRealxS → S, where S is the set of all subsets of E^3, returns Rotate(a,b,c,u) = v where:

$$v = \{q \mid p \in u \text{ and } [X_q, Y_q, Z_q] = [X_p, Y_p, Z_p] \times M_R\}$$

X, Y, and Z represent the coordinate values of their subscript point, and M_R is the three dimensional rotation matrix performing a rotation around the X-axis by angle c, followed by a rotation around the Y-axis by angle b, follwed by a rotation around the Z-axis by angle a [GIL78]:

$$M_R = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$

m_{11} = cos a.cos b

m_{12} = sin a.cos b

m_{13} = -sin b

m_{21} = -sin a.cos c + cos a.sin b.sin c

m_{22} = cos a.cos c + sin a.sin b.sin c

m_{23} = cos b.sin c

m_{31} = sin a.sin c + cos a.sin b.cos c

m_{32} = −cos a.sin c + sin a.sin b.cos c

m_{33} = cos b.cos c

$\square$

REFERENCES

[AGI76] Agin, G. J. and Binford, T. O. "Computer Description of Curved Objects", IEEE Trans. on Computers C-25, 4, April 1976.

[AGS78] "AGS880 Specification, 3D Graphic Operating System", Applicon Inc., 1978.

[ALL82] Allen, G. "Future Trends in Geometric Modeling", Proc. Conf. on CAD/CAM Technology in Mechanical Engineering, MIT, March 1982.

[APP76] Appel, A. and Will, P. M. "Determining the Three-dimensional Convex Hull of a Polyhedron", IBM J. Res. Dev. 20, 6, November 1976.

[APT80] "Computer-Graphics Augmented Design and Manufacturing System APT Interface Module Installation and Programmers Guide", IBM SH20-2095, May 1980.

[ARB81] Arbab, F., Melkanoff, M. A., and Morgan, M. E. "CAD/CAM Concepts and Facilities Lecture Notes", HyTek CAD/CAM Inc., 1981.

[ARB82a] Arbab, F., Cantor, D. G., Lichten, L., and Melkanoff, M. A. "The MARS CAM-Oriented Modeling System", Proc. Conf. on CAD/CAM Technology in Mechanical Engineering, MIT, March 1982.

[ARB82b] Arbab, F., Lichten, L., and Melkanoff, M. A. "Toward CAM-Oriented CAD", Proc. 19th Design Automation Conf., June 1982.

[BAE78] Baer, A. "Maintaining Integrity in Complex Shape Definitions", Proc. 15th Design Automation Conf., June 1978.

[BAE79] Baer, A., Eastman, C., and Henrion, M. "Geometric Modeling: A Survey", Computer Aided Design, 11, 5, September 1979.

[BAR82] Barsky, B. A. and Fournier, A. "Computational Techniques for Parametric Curves and Surfaces", Proc. Graphics Interface '82, NCGA of Canada/Canadian Man-Computer Communication Society, National Research Council of Canada, May 1982.

[BEZ82] Bezier, P. E. "A Computer Aided System for Car Body Design", Proc. Graphics Interface '82, NCGA of Canada/Canadian Man-Computer Communication Society, National Research Council of Canada, May 1982.

[BOO63] Booker, P. J. "A History of Engineering Drawing", Chatto and Windus, 1963.

[BOY79a] Boyse, J. W. "Interference Detection Among Solids and Surfaces", CACM 22, 1, January 1979.

[BOY79b] Boyse, J. W. "Data Structure for a Solid Modeller", GMR-2933, General Motors Research Labs, March 1979.

[BOY80] Boyse, J. W. and Rosen, J. M. "GMSOLID--A System for Interactive Design and Analysis of Solids", GMR-3451, General Motors Research Labs, October 1980.

[BOY82] Boyse, J. W. and Gilchrist, J. E. "GMSolid: Interactive Modeling for Design and Analysis of Solids", Computer Graphics and Applications 2, 2, March 1982.

[BRA75a] Braid, I. C. "The Synthesis of Solids Bounded by Many Faces", CACM 18, 4, April 1975.

[BRA75b] Braid, I. C. "Six Systems for Shape Design and Representation - a review", CAD Group Document No. 87, University of Cambridge, May 1975.

[BRA78] Braid, I. C. "New Directions in Geometric Modeling", Proc. Geometric Modeling Proj. MTG, CAM-I, Inc. St. Louis, MO, March 1978.

[BRO78] Brown, C. M., Requicha, A. A. G., and Voelcker, H. B. "Geometric Modeling Systems for Mechanical Design and Manufacturing", Proc. ACM Annual Conf., December 1978.

[BRO82] Brown, C. M. "PADL-2: A Technical Summary", Computer Graphics and Applications 2, 2, March 1982.

[BUR75] Burkley, R. M. and Mayfield, J. P. "A Numerical Geometry System", Technical Paper MS75-723, Society of Manufacturing Engineers, 1975.

[CAD80] "CADAM Training Manual", Lockheed-California Company, March 1980; also IBM SH20-2035.

[CAR78] Carlbom, I. and Paciorek, J. "Planar Geometric Projections and Viewing Transformations", Computing Surveys 10, 4, December 1978.

[COS82] Cosmai, G., Cugini, U., Mussio, P., and Napolitano, A. "An Interactive Drafting System Based on Two Dimensional Primitives", Proc. 19th Design Automation Conf., June 1982.

[COU65] Courant, R. and John, F. "Introduction to Calculus and Analysis", Vol. 1, John Wiley and Sons, 1965.

[DAD82] Dadoun, N. and Kirkpatrick, D. G. "Hierarchical Approaches to Hidden Surface Intersection Testing", Proc. Graphics Interface '82, NCGA of Canada/Canadian Man-Computer Communication Society, National Research Council of Canada, May 1982.

[DDM80] "Calma DDM", General Electric Calma Company, Sunnyvale, CA 1980.

[DVS82] "The Computervision Designer V System", Computervision Corp., Bedford, MA, 1982.

[EAS79] Eastman, C. and Weiler, K. "Geometric Modeling Using the Euler Operators", Proc. 1st Annual Conf. on Computer Graphics in CAD/CAM Systems, MIT, April 1979.

[EAS80] Eastman, C. M. and Henrion, M. "The GLIDE Language for CAD", Journal of the Technical Councils of ASCE 106, TC1, August 1980.

[EAS77] Eastman, C. M. and Henrion, M. "GLIDE: A Language for Design Information Systems", Computer Graphics, 11, 2, July 1977.

[EDM80] "Encyclopedic Dictionary of Mathematics", Mathematical Society of Japan, MIT Press, 1980.

[FAU79] Faux, I. D. and Pratt, M. J. "Computational Geometry for Design and Manufacture", Ellis Horwood Ltd., 1979.

[FEN80] Feng, D. Y. and Riesenfeld, R. F. "Some New Surface Forms for Computer Aided Geometric Design", Computer Journal 23, 4, November 1980.

[FIT81a] Fitzgerald, W., Gracer, F., and Wolfe, R. "GRIN: Interactive Graphics for Modeling Solids", IBM J. Res. Dev. 25, 4, July 1981.

[FIT81b] Fitzgerald, W. J. "Using Axial Dimensions to Determine Proportions of Line Drawings in Computer Graphics", Computer-Aided Design, 13, 6, November 1981.

[FIT82] Fitzgerald, W. J., Gracer, F., and Wolfe, R. N. "The IBM SOLID Modeling System User Guide", IBM T. J. Watson Research Center, March 1982.

[FOR72] Forrest, A. R. "Mathematical Principles for Curve and Surface Representation", Proc. Curved Surfaces Eng., IPC Science and Technology Press, 1972.

[FRA82] Franklin, W. R. "Efficient Polyhedron Intersection and Union", Proc. Graphics Interface '82, NCGA of Canada/Canadian Man-Computer Communication Society, National Research Council of Canada, May 1982.

[FRE58] French, T. E. and Vierck, C. J. "Graphic Science", McGraw-Hill, 1958.

[FRE66] French, T. E. and Vierck, C. J. "A Manual of Engineering Drawing for Students and Draftsmen", McGraw-Hill, 1966.

[GIE76] Giesecke, F. E., Mitchell, A., Spencer, H. C., and Hill, I. L. "Technical Drawing", Macmillan Company, 1976.

[GIL78] Giloi, W. K. "Interactive Computer Graphics", Prentice-Hall, 1978.

[GJO80] Gjovaag, J. and Perry, R. N. "Drafting as a Paradigm for the User Interface to a Geometric Modeler", Proc. AUTOFACT WEST, Anaheim, CA, November 1980.

[GOL79] Goldstein, R. and Malin, L. "3D Modeling with the Synthavision System", Proc. 1st Annual Conf. on Computer Graphics in CAD/CAM Systems, MIT, April 1979.

[GOP79] Gopin, A. and Gossard, D. C. "Symbolic Dimensioning", Proc. 1st Annual Conf. on Computer Graphics in CAD/CAM Systems, MIT, April 1979.

[GOR74a] Gordon, W. J. and Riesenfeld, R. F. "B-spline Curves and Surfaces" in Computer Aided Geometric Design, R. E. Barnhill and R. F. Riesenfeld (eds.), Academic Press, 1974.

[GOR74b] Gordon, W. J. and Riesenfeld, R. F. "Berstein-Bezier Methods for the Computer-Aided Design of Freedom Curves and Surfaces", JACM 21, 2, April 1974.

[GOS80] Gossard, D. J. "A CAD/CAM Story", in The CAD/CAM Handbook, Computervision Corp., 1980.

[GRA80] "GRAF: UNIGRAPHICS Usera Operational Description", McDonnel Douglas Automation Company, May 1980.

[GRI80] "GRIP: UNIGRAPHICS Interactive Programming Manual", McDonnel Douglas Automation Company, 1980.

[GRO76] Grossman, D. D. "Procedural Representation of Three-dimensional Objects", IBM J. Res. Dev. 20, 6, November 1976.

[GTA79] "Westinghouse Electric Corporation Graphic Training Aids Systems Guide", IBM LY20-9004, 1979.

[HAR82] Hardt, D. E. "CAD/CAM for Sheet Metal Forming", Proc. Conf. on CAD/CAM Technology in Mechanical Engineering, MIT, March 1982.

[HIL78a] Hillyard, R. "Dimensions and Tolerances in Shape Design", Tech. Rep. No. 8, Computer-aided Design Group, University of Cambridge, June 1978.

[HIL78b] Hillyard, R. and Braid, I. C. "Analysis of Dimensions and Tolerances in Computer-Aided Mechanical Design", Computer-Aided Design, 10, 3, June 1978.

[HIL78c] Hillyard, R. and Braid, I. C. "Characterizing Non-Ideal Shapes in Terms of Dimensions and Tolerances", Computer Graphics 12, 3, August 1978.

[HIL82a] Hillyard, R. "The Build Group of Solid Modelers", Computer Graphics and Applications 2, 2, March 1982.

[HIL82b] Hillyard, R. "A Taxonomy of Engineering Geometric Modeling System", Proc. Conf. on CAD/CAM Technology in Mechanical Engineering, MIT, March 1982.

[INO82] Inoue, K., Adachi, M., and Funayama, T. "A Layout System for High Precision design of Progressive Die", Proc. 19th Design Automation Conf., June 1982.

[IUE82] "User Exit Applications", Release IUER104, Lockheed-California Company, 1982.

[JOY81] Joyce, J. D. General Motors Research Laboratories, Warren, Michigan, Personal communication, September 1981.

[KAK76] Kakino, Y., et al "A New Method of Parts Description for Computer-Aided Production Planning", Proc. PROLAMAT 76, June 1976.

[KAL82] KALAY, Y. E. "Modeling Polyhedral Solids Bounded by Multi-Curved Parametric Surfaces", Proc. 19th Design Automation Conf., June 1982.

[KAM83] Kamvar, E. "Recognition and Processing of Graphical Data for Manufacturing Engineering", Ph.D. Dissertation in preparation, University of California, Los Angeles, 1983.

[KOS79] Kossuth, G. J., Hare, R. L., and Dunlevey, M. "Input and Output Modes for the SHAPES System for Computer-Aided Mechanical Design", Proc. 1st Annual Conf. on Computer Graphics in CAD/CAM Systems, MIT, April 1979.

[LAN79] Lanning, J. H. and Madden, S. J. "Capabilities of the SHAPES System for Computer-Aided Mechanical Design", Proc. 1st Annual Conf. on Computer Graphics on CAD/CAM Systems, MIT, April 1979.

[LAN80] Lane, J. M., Carpenter, L. C., Whitted, T., and Blinn, J. F. "Scan Line Methods for Displaying Parametrically Defined Surfaces", CACM 23, 1, January 1980.

[LIG82a] Light, R. A. "Variational Geometry: Modification of Part Geometry by Changing Dimensional Values", Proc. Conf. on CAD/CAM Technology in Mechanical Engineering, MIT, March 1982.

[LIG82a] Light, R. A. and Gossard, D. C. "Modification of Geometric Models through Variational Geometry", Computer-Aided Design, 14, 4, July 1982.

[LIN81] Lin, V. C., Gossard, D. C., and Light, R. A. "Variational Geometry in Computer-Aided Design", Computer Graphics, 15, 3, August 1981.

[LIU82] Liu, C. K. and Eastman, C. M. "Design of a Graphic Processor for Computer-Aided Drafting", Proc. 19th Design Automation Conf., June 1982.

[LUZ59] Luzadder, W. J. "<u>Fundamentals of Engineering Drawing</u>", Prentice-Hall, 1959.

[MAR80] Markowsky, G. and Wesley, M. A. "Fleshing out Wireframes", IBM J. Res. Dev. 24, 5, September 1980.

[MAY76] Mayfield, J. P. and Burkley, R. M. "Applications of a Numerical Geometry System in Engineering", Proc. 13th Design Automation Conf., June 1976.

[MEL82a] Melkanoff, M. A., Lichten, L., and Arbab, F. "An Overview of CAD/CAM with Special Emphasis on Mechanical Applications", UCLA Computer Science Department Quarterly, 10, 1, Winter 1982.

[MEL82b] Melkanoff, M. A., Arbab, F., Lichten, L., and Morgan, M. E. "CAD/CAM Lecture Notes and Laboratory Problems", E194A/B Course Notes, Computes Science Department, UCLA 1982.

[MER82] Merritt C. M. "Curves for Modeling Chromosome Shapes", Proc. Graphics Interface '82, NCGA of Canada/Canadian Man-Computer Communication Society, National Research Council of Canada, May 1982.

[MES79] "Computer-Graphics Augmented Design and Manufacturing System 3D Mesh Geometry Supplement to User Training Manual", IBM SH20-2181, May 1979.

[MYE82] Myers, W. "An Industrial Perspective on Solid Modeling", Computer Graphics and Applications 2, 2, March 1982.

[NEW79] Newman, W. M. and Sproull, R. F. "Principles of Interactive Computer Graphics", McGraw-Hill, 1979.

[NGS80] "Numerical Geometry System", IBM SB21-2610-1, 1980.

[OKI73] Okino, N., Kakazu, Y., and Kubo, H. "TIPS-1, Tecnical Information Processing System", Computer Languages for Numerical Control, pp 141-150, North-Holland, 1973.

[OKI76] Okino, N., Kubo, H., and Kakazu, Y. "TIPS-2: An Integrated CAD/CAM System", Proc. PROLAMAT 76, June 1976.

[PAD74] "An Introduction to PADL: Characteristics, Status, and Rationale", Production Automation Project TM-22, University of Rochester, December 1974.

[PAR53] Parkinson, A. C. "Pictorial Drawing For Engineers", Pitman and Sons, 1953.

[PRE81] Preiss, K. "Algorithms for Automatic Conversion of a 3-View Drawing of a Plane-Faced Part to 3-D Representation", Computers in Industry 2, pp 133-139, North-Holland, 1981.

[REQ77a] Requicha, A. A. G. "Part and Assembly Description Languages: I- Dimensioning and Tolerancing", Production Automation Project TM-19, University of Rochester, May 1977.

[REQ77b] Requicha, A. A. G. and Voelcker, H. B. "Constructive Solid Geometry", Production Automation Project TM-25, University of Rochester, November 1977.

[REQ77c] Requicha, A. A. G. "Mathematical Models of Rigid Solid Objects", Production Automation Project TM-28, University of Rochester, November 1977.

[REQ78] Requicha, A. A. G. and Tilove, R. B. "Mathematical Foundations of Constructive Solid Geometry: General Topology of Closed Regular Sets", Production Automation Project TM-27, University of Rochester, November 1978.

[REQ80a] Requicha, A. A. G. "Representation of Rigid Solid Objects", Chapter 1 of Computer Aided Design, Modelling, Systems Engineering, CAD-Systems, Lecture Notes in Computer Science 89, Springer-Verlag, 1980.

[REQ80b] Requicha, A. A. G. "Representation of Rigid Solid Objects", Production Automation Project TM-29, University of Rochester, June 1981.

[REQ80c] Requicha, A. A. G. "Representations for Rigid Solids: Theory, Methods, and Systems", Computing Surveys 12, 4, December 1980.

[REQ81] Requicha, A. A. G. and Voelcker, H. B. "Geometric Modelling of Mechanical Parts and Machining Processes", SIGGRAPH 81 Seminar on Solid Modeling, August 1981.

[REQ81] Requicha, A. A. G. and Voelcker, H. B. "An Introduction to Geometric Modeling and its Applications in Mechanical Design and Production", Chapter 5 of Advances in Information Systems Science, 8, J. T. Tou (ed.), Plenum Press, 1981.

[REQ82] Requicha, A. A. G. and Voelcker, H. B. "Solid Modeling: A Historical Summary and Contemporary Assessment", Computer Graphics and Applications 2, 2, March 1982.

[ROT80] Roth, S. D. "Ray Casting as a Method for Solid Modeling", GMR-3466, General Motors Research Labs, October 1980.

[ROT82] Roth, S. D. "Ray Casting for Modeling Solids", Computer Graphics and Image Processing, 18, pp 109-144, Academic Press, 1982.

[SAM76] Samuel, N. M., Requicha, A. A. G., and Elkind, S. A. "Methodology and Results of an Industrial Part Survey", Production Automation Project TM-21, University of Rochester, July 1976.

[SIL81] Silva, C. E. "Alternative Definitions of Faces in Boundary Representations of Solid Objects", Production Automation Project TM-36, University of Rochester, November 1981.

[SIM63] Simmons, G. F. "Topology and Modern Analysis", McGraw-Hill, 1963.

[SUR79] "Computer-Graphics Augmented Design and Manufacturing System 3D Surface Geometry Supplement to User Training Manual", IBM SH20-2182, May 1979.

[TOW30] Townsend, C. E. and Cleary, S. F. "Introductory Mechanical Drawing", John Wiley and Sons, 1930.

[TUR50] Turner, W. W., Buck, C. P., and Ackert, H. P. "Basic Engineering Drawing", Ronald Press, 1950.

[VEE82] Veeman, P. "The design of Sculptured surfaces using Recursive Subdivision Techniques", Proc. Conf. on CAD/CAM Technology in Mechanical Engineering, MIT, March 1982.

[VOE78] Voelcker, H. B., et al "The PADL-1.0/2 System for defining and displaying solid objects", Computer Graphics (Proc. Siggraph '78), 12, 3, August 1978.

[VOE81] Voelcker, H. B. and Hunt, W. "The Role of Solid Modelling in Machining-Process Modelling and NC Verification", Proc. SAE Int. Congress and Exhibit., February 1981.

[WES80a] Wesley, M. A., et al "A Geometric Modelling System for Automated Mechanical Assembly", IBM J. Res. Dev. 24, 1, January 1980.

[WES80b] Wesley, M. A. "Construction and Use of Geometric Models", Chapter 2 of Computer Aided Design, Modelling, Systems Engineering, CAD-Systems, Lecture Notes in Computer Science 89, Springer-Verlag, 1980.

[WES81] Wesley, M. A. and Markowsky, G. "Fleshing Out Projections", IBM J. Res. Dev. 25, 6, November 1981.

[WOO76] Woo, T. C. "Computer Aided Recognition of Volumetric Design", Proc. PROLAMAT 76, June 1976.

[WOO82] Woo, T. C. "Feature Extraction by Volume Decomposition", Proc. Conf. on CAD/CAM Technology in Mechanical Engineering, MIT, March 1982.