

Functional Requirements to Shape Generation in CAD

By

Jinpyung Chung

B.S. Mechanical Engineering, Seoul National University, 1991
M.S. Mechanical Engineering, Seoul National University, 1996

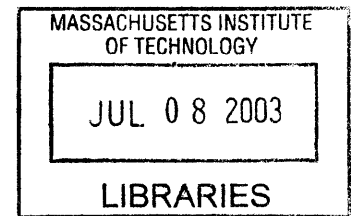
Submitted to the Department of Mechanical Engineering
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Mechanical Engineering

at the

Massachusetts Institute of Technology

June 2003

© 2002 Massachusetts Institute of Technology
All rights reserved



Signature of Author_____

A handwritten signature in black ink, appearing to be "Jinpyung Chung", written over a horizontal line.

Department of Mechanical Engineering
May 23, 2003

A handwritten signature in black ink, appearing to be "Nam P. Suh", written over a horizontal line.

Certified by_____

Nam P. Suh
Ralph E. & Eloise F. Cross Professor of Mechanical Engineering
Thesis Supervisor

Accepted by_____

Ain A. Sonin
Chairman, Department Committee on Graduate Students

BARKER

Functional Requirements to Shape Generation in CAD

by

Jinpyung Chung

Submitted to the Department of Mechanical Engineering
on May 23, 2003 in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy in
Mechanical Engineering

Abstract

An outstanding issue in computer-aided design (CAD) is the creation of geometric shapes from the description of functional requirements (FRs). This thesis presents a method that can generate assembled shapes from the given FRs without human intervention. To achieve this goal, the design process follows a V-model of decomposition and integration based on axiomatic design. The V-model consists of three main sub-processes; (1) a top-down decomposition of FRs and design parameters (DPs), (2) mapping of DPs into geometric entities, and (3) a bottom-up integration of the geometric entities. A shape decomposition technique is used in the V-model to generate solid cells from the geometric entities in the CAD models based on FRs. These cells are stored and reused during the integration process. A set of cells mapped to an FR is called a functional geometric feature (FGF) to differentiate it from geometric features defined by only geometric characteristics. Each FGF has mating faces as its pre-defined interfaces. Links of FR-DP-FGF-INTERFACES and their hierarchies are made and stored in the database as fundamental units for automatic assembled shape generation. The retrieval of proper FGF from the database is performed by matching a query FR with stored FRs by a lexical search based on the frequency of words and the sequence of the words in the FR statements using a synonym checking system. The language-matching rate is calculated as a value of FR_metric between 0 and 1. A computer algorithm automatically combines and assembles the retrieved FGFs. Genetic algorithm (GA) searches for the best combination for matching interface types and generates assembly sequences. From the highest-valued chromosome, the computer algorithm automatically assembles FGFs by coordinating, orienting, and positioning with reference to the given mating conditions and calculates geometric interface-ability to a value of INTERFACE_metric between 0 and 1. The higher the values of FR_metric and INTERFACE_metric, the better the generated design solution for the given FRs that must be satisfied. The process of top-down decomposition and bottom-up integration reduces the number of possible combinations of interfacing FGFs. Design matrix visually relates FRs to FGFs. The method presented in this thesis has demonstrated that a "functional CAD" can aid designers in generating conceptual design

solutions from functional descriptions, in reusing existing CAD models, and in creating new designs.

Keywords: CAD (computer-aided design), axiomatic design theory, design process, top-down, bottom-up, FR-DP-FGF-INTERFACES, database, language matching, assembled shape generation, interface-ability

Thesis Supervisor: Nam P. Suh

Title: Ralph E. & Eloise F. Cross Professor of Mechanical Engineering

Acknowledgments

I thank God for giving me wisdom to finish my PhD thesis. He made me lie down in green pastures, and leaded me beside quiet waters, whenever I met difficulties at MIT. I also give thanks to my family. My wife has been one of the greatest supporters in my life. She has provided constant encouragement, prayer and love throughout my academic career, taking care of three children. My parents have endured me while I have been studying for a long time. I confess that I could not finish my PhD thesis without their support.

I feel very fortunate to have had Professor Nam P. Suh as my thesis advisor and mentor. He questioned me a lot of intellectual challenges, and also advised me to be able to answer them. He picked my thesis topic, “Thinking Design Machine (TDM),” and has guided me to make progress in implementing TDM, which had looked difficult to me. Frequently, I was surprised at his capturing my hazy reasoning and pointing out the crucial factors to solve problems. I believe that he must be a person in respect in the field of engineering over the world. He was also patient to improve the way I use English, still that I am not fluent in. I truly thank Professor Suh for being my advisor and mentor. I am also grateful to the other members of my thesis committee for their precious time and advice. Professor David C. Gossard gave me important hints to make my thesis valuable. Professor Sanjay E. Sarma made me ponder through his sharp questions and comments.

I am grateful to all the great members in Axiomatic Design Group. Dr. Tae-Sik Lee was my office-mate, who has spent four years with me. We have had frequent discussions about axiomatic design. He provided me with many pointers important to my research. Dr. Jason Melvin was a cooperative helper for my presentations. He made a lot of useful suggestions for my presentations. Hrishikesh Deo did not hesitate to proofread my abstract, and has organized our group meetings for us, including me. Dr. Rajesh Jugulum was an expert on robust design. I really enjoyed talking on the topic with him.

When I needed friendship, these friends have been always beside me and cheered me up to refresh my strength and energy. I thank Sangjun Han, Hyungsoo Moon, Yun Kang, Joonil Seog, Hyuksang Kwon, Daekeun Kim, and Seungkil Son for their friendship. I also thank Pastor Joseph Choi and Dr. Daesung Choi for their prayers for me.

Finally, I am very grateful to Professor Kunwoo Lee, who was my advisor during my Master’s study in Seoul National University. I could not have studied at MIT, if it had not been for his advice. He introduced fundamentals of CAD to me, which have been an important base to reach this point.

**This thesis is dedicated
To my wife, Sookyung Lee and
To my father Dr. Joyoung Chung.**

Table of Contents

Abstract	3
Acknowledgments	7
Table of Contents	9
List of Figures	11
List of Tables	13
1 Introduction	15
1.1 Problems of Current CAD Systems	16
1.2 Feature-Based Design	16
1.3 Aim of Thesis	18
1.4 Overview of A Proposed Method	18
1.5 A Limitation and Expected Impact	21
2 Axiomatic Design Framework	23
3 Design Process	27
3.1 V-Model	27
3.2 Illustrative Examples of V-model	31
3.2.1 Pawl and Cam Mechanism Design	32
3.2.2 Hanger Design	39
4 Database	43
4.1 System Architecture	43
4.2 Data Structure	45
4.3 Functional Geometric Feature (FGF)	46
5 Generation of Design Solutions	61
5.1 Language Matching of Functional Requirements	61
5.2 Assembled Shape Generation	63
5.2.1 Combinations of FGFs	63
5.2.2 Automatic Assembly and Interface-ability of FGFs	69
5.3 Top-down Decomposition for Combining FGFs	89
5.4 Shape Determination	91
6 Application Examples	93
6.1 Assembled Shape Generation of two FGFs	94
6.2 Addition of a New FR to a Complete Design	102
6.3 Integration of FGFs for Beverage Can Design	106
6.4 Inference of FRs from Interfaces	110

7 Conclusions	113
7.1 Concluding Remarks	113
7.2 Contributions	114
7.3 Vision and Future Work	117
References	119

List of Figures

Figure 1. Zigzagging decomposition/mapping/composition process of axiomatic design	23
Figure 2. Design Range (DR) and System Range (SR)	25
Figure 3. Axiomatic thinking process for designing physical products	29
Figure 4. Fundamental structure of decomposed FRs, DPs, and geometric entities	31
Figure 5. Full design matrix of pawl-cam design	33
Figure 6. Possible alternatives based on topology construction of the pawl-cam design	36
Figure 7. Links of FRs to geometric entities based on the design matrix for alternative 1	38
Figure 8. DPs and dimensions of alternative 1	39
Figure 9. Full design matrix of the hanger design	40
Figure 10. Links of FRs to geometric entities based on the design matrix for alternative	42
Figure 11. System architecture and flow chart	44
Figure 12. Fundamental structure of database	45
Figure 13. Operations: cut and subtraction	48
Figure 14. Graph representations of FGFs	51
Figure 15. Data structure between FGFs and cells for beverage can	56
Figure 16. Data structure between FGFs and cells of cam and pawl mechanism	58
Figure 17. Data structure between FGFs and cells of hanger design	59
Figure 18. Possible combinations of connectivity of three FGFs in the same level	65
Figure 19. Connections and assembly sequence between FGFs	66
Figure 20. Processes to check interface-ability between FGFs	69
Figure 21. Shape integration process to check geometric interface-ability	71
Figure 22. Orienting of the interfacing FGF	79
Figure 23. Calculation of AF_metric	80
Figure 24. Possible combinations of connectivity of three FGFs in a hierarchical tree	90
Figure 25. Implementation of FR, DP trees, language matching and GA	94
Figure 26. A simple example for search of proper FGFs and combination of the FGFs	95
Figure 27. Encoding in GA and calculation of IT_metric	96

Figure 28. Initial configuration of FGFs and their geometric data _____	98
Figure 29. Results of shape generation from two FRs _____	99
Figure 30. Assembled shapes that chromosomes present _____	101
Figure 31. Final results of container design problem _____	101
Figure 32. Addition of FGF2 to the pawl _____	102
Figure 33. Results of interfacing a rib to the pawl _____	104
Figure 34. Resultant assembled shape for additional FR to pawl and cam design _____	105
Figure 35. Definition of FRs and the corresponding candidate FGFs _____	106
Figure 36. GA codes of mating faces and an example chromosome _____	107
Figure 37. Resultant shapes for the beverage can design problem _____	109
Figure 38. Advantage of top-down decomposition _____	110
Figure 39. Example for inferring FR and DP from interfaces _____	111

List of Tables

Table 1. Geometric entities mapped from leaf level DPs of the pawl-cam design _____	34
Table 2. Defined interfaces and corresponding DPs of the pawl and cam design _____	35
Table 3. Geometric entities mapped from leaf level DPs of the hanger design _____	40
Table 4. Defined interfaces and corresponding DPs of the hanger design _____	41
Table 5. Design alternatives of the hanger design _____	41
Table 6. FRs, DPs and the corresponding FGFs of beverage can _____	54
Table 7. FRs, DPs and the corresponding FGFs of cam and pawl mechanism _____	57
Table 8. FRs, DPs and the corresponding FGFs of hanger design _____	59
Table 9. Encoding of interfaces of FGFs into genes _____	67
Table 10. Combinations of interfacing faces represented by a chromosome (2 0 9 17 18 6) _____	68
Table 11. Combinations of interfacing mating faces in a chromosome _____	72
Table 12. Assembly methods and their output DOFs _____	86

Chapter 1

Introduction

Computer has advanced the field of design. The most significant use of computers has been in representing and manipulating geometry. However, the idea that we will use computers to generate design solutions that exceed human capability still remains to be an intellectual challenge. The challenge is even greater if we are to design a complex systems such as the Orbital Space Plane using the ability of computers to go from functional description of the design goals to geometric shapes. To achieve the next stage of advances in utilizing computers in mechanical design, computers should be used to generate geometric shapes that satisfy a given set of functional requirements (FRs) so as to generate creative design solutions that defy human imagination.

Design process may be divided into three stages. Stage 1 is a mapping from customer needs to functional requirements (FRs) where information on the product is collected and transformed into functional requirements. Stage 2 is the design of the artifacts that satisfy the FRs. In many mechanical design situations, it involves the generation of geometric embodiments that satisfy the FRs. These high-level designs (some call it conceptual design) gradually acquire design details as we decompose the FRs and design parameters, (DPs). In the final stage, which may be called the leaf-level design, dimensions and geometric shapes of individual components, are determined in mechanical design. The high-level design decisions, plays an important role in the overall success of design. Unfortunately, design is done empirically, which involves iteration of the “design/build/test” cycles until the design seems to work. This empirical approach to design is costly and unreliable. The goal of any design

process ought to be “design it right first time”. Otherwise, the performance, reliability, and robustness of large complicated systems cannot be assured at the design stage. The empirical process may be responsible for the cost over-run and the long development time.

1.1 Problems of Current CAD Systems

Current CAD technology successfully supports generation of geometric shapes and assemblies of the designed artifacts in detailed design stage using solid modeling techniques and specific data structures. However, current CAD programs are understood as drawing packages, which cannot be used to support the conceptual design process. Since commercially available CAD systems cannot reason and create geometric shapes based on the FRs of the design tasks, all the basic design decisions on geometric details must be made by human designers. Solid modeling techniques are useful only after detailed shapes of the designed artifacts have been determined. The lack of the functional representation in CAD models limits the usefulness of CAD in developing a new product, because designers do not understand why certain shapes are created, and must update the CAD models based only on the geometric shapes and their assemblies. Furthermore, it is difficult to reuse existing design knowledge or concept using existing CAD models in developing new geometric artifacts with computer support. In most CAD packages, the designed artifacts are decomposed only in the physical domain, relating only information and knowledge of geometry of the artifacts. However, the artifacts are designed to satisfy FRs and therefore, they must be functionally decomposed from the early stage of the design. Thus, information and knowledge of functional requirements should be related to those of geometric artifacts in CAD systems.

1.2 Feature-Based Design

The organization of design information is one of the most important factors that determine performance and functionality of the CAD systems. Most CAD systems are primarily concerned with data structure and information flow for representing and

constructing 3D shapes on computers. Those CAD systems enable us to generate complex shapes. They also utilize the geometric information in designing and manufacturing products. Feature-based representation or design is one of the well-known results of such an effort. The concept of *design by features* was first proposed by Pratt and Wilson. [1] It seems to be conceptually agreed that one of the key benefits using features is that the designers can put their design intent into the geometric model. Also, many techniques using features have been developed to automate machining, to diagnose the defect of geometric models, to generate finite element models, and so on.

Nevertheless, there still exist two main bottlenecks for the feature-based design systems to be useful for designing geometric artifacts. One is the lack of the means of defining features based on functions, though a major benefit of using features is to put design intent into geometry. The other is difficulty to collect geometric features to be reused. The problems exist because most research on features or geometric models is conducted only in the physical domain. For example, Dixon and Cunningham [2] proposed a system for *design with features*, in which they classified static features into five types; primitives, intersections, add-ons, macros, and whole-forms, and then constructed a feature library to reuse them. Sreevalsan and Shah [3] addressed the problem of integrating *design by features*, interactive definition, and automatic recognition. They provided designers with templates to edit features by formal language. Their taxonomy is useful to handle geometric features in a physical domain, but no information on functions of the features is included in their design system. In addition, capability of the feature based design system is limited by the number of the collected features or templates, but it is not easy to collect the pre-defined features for design intent only in terms of geometric features.

There exist many CAD approaches to using database. Most of them classify CAD models by similarity of the shape, stored them in the database, and searched similar shape to an input geometric model. In these approaches, it is relatively easy to collect and classify CAD models and to find out similar shapes, but this database cannot help designers to

generate design solutions. It is usually used to find out similar parts for manufacturing purpose.

1.3 Aim of Thesis

The problems described in the previous paragraphs can be summarized as the following questions. The aim of this thesis is to find out a systematic method by which the questions can be answered.

- How can we relate the geometric features of the CAD models to functional requirements?
- How can a computer reuse the geometric features to generate candidate shapes in the conceptual design phase?

To answer these two questions, the following five sub-topics have been considered and will be explained in the following sections of this thesis.

- Design process
- Shape decomposition to geometric features
- Data structure
- Language matching
- Automatic assembly of the geometric features

1.4 Overview of A Proposed Method

The methodology presented in this thesis uses axiomatic design theory to provide a framework in implementing an intelligent CAD program to support conceptual design tasks. Axiomatic design framework [4, 5] has been created to incorporate functional aspects of the designed artifacts based on a natural thinking process for conceptual design. We extend the

axiomatic design theory to create a formalized design process, called V-model, for designing geometric artifacts using computer aid. The goal of the design process is to generate the links between functions, actual language descriptions of the functions, i.e. FRs, and geometric entities in a most logical way.

Xue et al. [6] implemented III (Intelligent Integrated Interactive) CAD system to support conceptual design. It was a rule-based expert system, which has relational database of physical principles and objects. It may have a limitation to store design knowledge and to extract proper design knowledge, because it's not easy to collect all the physical principles and objects and create rules. In contrast to this, we used case-based approach. [7] We are trying to collect as many design cases as possible through V-model design process, and to store them as properly formatted information into a database. The database has information on language descriptions of FRs and DPs, and decomposed geometric features mapped to the FRs and DPs. The information is related to how they are decomposed in the hierarchical trees and mapped together. The larger the database, the more helpful to search design solutions as empirically proved in the CYC project. [8] In CYC project, many knowledge engineers have been collecting common sense knowledge and organizing them through top-down decomposition in knowledge bases for about 15 years. It now supports language understanding and infer relevant knowledge from the input queries, though it cannot perform real thinking to generate solutions.

In this thesis, cell decomposition technique has been used to decompose a whole solid or an assembled CAD models into a set of geometric features. Sakurai and Gossard [9] developed an algorithm to automatically recognize geometric features such as pockets and to delete unnecessary geometric features, and Sakurai [10] found out a set of maximal convex cells from concave edges, a combination of which can be mapped to a sequence of machining. Several cell decomposition techniques have been used to decompose a whole solid into basic cells to automate manufacturing process from CAD models [11-13] or generate finite elements [14]. Most shape decompositions using cell decomposition technique are performed to find out geometric features from characteristics of input geometry. Our shape

decomposition is performed to separate the geometric features from a whole solid or an assembly based on FRs. The decomposed geometric features by FRs are called functional geometric features(FGFs) hereafter in this thesis, because it is defined by functional requirements rather than geometric characteristics. During shape decomposition process, information on interfaces of the FGFs can be extracted and stored in the database. The traditional cell decomposition techniques gave us an idea, that is, a solid can be decomposed into a set of cells based on the geometric characteristics. If a set of cells can be related to an FR, the set is defined as an FGF to satisfy the corresponding FR.

Links of FR-DP-FGF-INTERFACES and their hierarchies are made and stored in the database.. Each link of FR-DP-FGF-INTERFACES is a fundamental unit for automatically generating assembled shapes from a given set of FRs. Each FGF is a set of cells with information on internal interfaces, by which the cells have been interfaced to form the FGF, and on dangling interfaces, which are mating faces of the FGF to be interfaced later with other FGFs. Detailed geometric information of each FGF has been encapsulated. Only information on dangling interfaces, i.e. mating faces, has been explicitly described and used for computer algorithms to automatically assemble the FGFs. The internal interfaces can be used to visualize the generated shapes.

The retrieval of proper FGF from the database is performed by matching a query FR with stored FRs by a lexical search based on the frequency of words and the sequence of the words in the FR statements using a synonym checking system. The language-matching rate is calculated to a value of FR_metric between 0 and 1. This algorithm is relatively simple comparing to other text-based knowledge base, because focus of this thesis is not text-based learning or knowledge extraction [15]. It only checks similarity between FR statements based on the words.

We developed a computer algorithm that automatically combines and assembles the retrieved FGFs. Genetic algorithm (GA) [16] searches for the best combination for matching the geometric interface and generates assembly sequences. From the highest-valued

chromosome, the computer algorithm automatically assembles FGFs by coordinating, orienting, and positioning with reference to the given mating conditions and calculates geometric interface-ability to a value of INTERFACE_metric between 0 and 1. The INTERFACE_metric is multiplications of three metrics for interface types, relative orientations between mating faces, and possibility to position mating faces.

The higher the values of FR-metric and INTERFACE_metric, the better the generated design solution for the given FRs that must be satisfied. The V-model design process of top-down decomposition and bottom-up integration reduces the number of possible combinations of interfacing FGFs.

1.5 A Limitation and Expected Impact

A limitation of our method is not to fully automate decomposing shapes by FRs. Because an FGF is not defined by geometric characteristics, but by FRs, it is not easy to automatically decompose solids in an assembly into cells and then to relate the cells to FGFs. We propose that the shape decomposition must be done by knowledge engineers guided by V-model design process.

Despite of the limitation of our method, our method can be a first step to answer the questions we raised as a break-through for common problems most CAD systems have. Our method provides designers with design environment for designing geometric artifacts from functional descriptions. A number of FRs, DPs, and FGFs can be related and collected in databases and they can be automatically assembled to generate candidate shapes from language descriptions of their functional requirements. It is estimated that there exist over 30 billion CAD models generated using commercial CAD packages in the world. [17] Most CAD models are topologically and geometrically almost complete and being standardized by STEP or IGES. However, it's not easy to reuse them for the conceptual design, because information on their functional intent has been missed so as to be impossible to understand why they are created in many design cases. The research on the automatic shape integration or

generation for reuse of design concepts is rare, because most research focuses on the complex geometry, which causes heavy computational load. In this thesis, we proposed a method. The proposed method can be a solution to reuse the CAD models by incorporating FRs into the CAD models in a manageable computational complexity. It is expected that our method may create shapes, that human designer did not think, by different combinations of pre-defined FGFs from given FRs. The following sections will be detailed explanations of our proposed method and application examples of the method.

Chapter 2

Axiomatic Design Theory

In axiomatic design theory, synthesized solutions that satisfy the highest-level FRs are created through a decomposition process that requires zigzagging between the functional domain and the physical domain as shown in Figure 1. It decomposes a top-level FR into leaf level FRs, which are not decomposable any further. Designer creates leaf level DPs in his brain or extracts those from his knowledge base to satisfy the corresponding FRs. Once leaf level DPs are found, they must be integrated to create the whole design artifact, which is then checked to determine if they work well and satisfy FRs based on two design axioms. The first axiom is the independence axiom, which states that the independence of FRs must be maintained in design. Second axiom is information axiom, which states that the information content must be minimized.

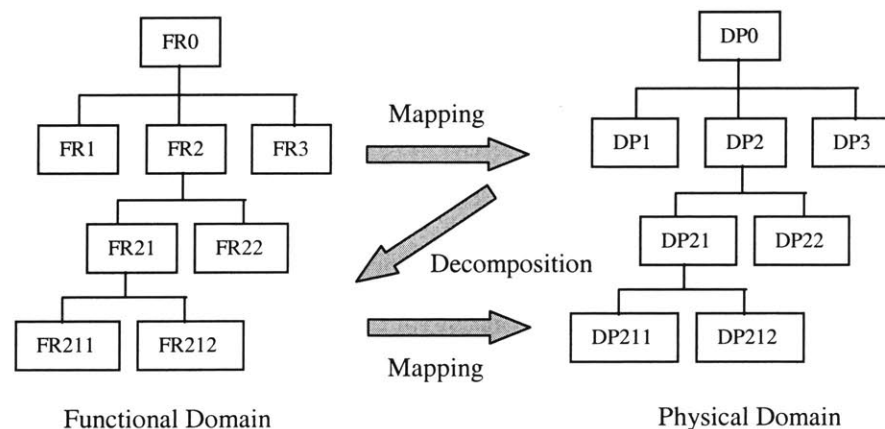


Figure 1. Zigzagging decomposition/mapping/composition process of axiomatic design

Mathematical representation of axiomatic design is

$$\begin{Bmatrix} FR1 \\ FR2 \\ FR3 \end{Bmatrix} = \begin{bmatrix} A11 & A12 & A13 \\ A21 & A22 & A23 \\ A31 & A32 & A33 \end{bmatrix} \begin{Bmatrix} DP1 \\ DP2 \\ DP3 \end{Bmatrix} \text{ where, } A_{ij} = \frac{\partial FR_i}{\partial DP_j}.$$

A_{ij} represents sensitivity, either qualitative or quantitative, of FR_i with respect to DP_j and it can be interpreted as a causality, which represents cause-effect between DPs and FRs. The elements A_{ij} of the design matrix (DM) are determined in mapping process. The following three matrices show coupling conditions represented by the design matrix.

$$\begin{bmatrix} X & O & O \\ O & X & O \\ O & O & X \end{bmatrix}$$

Uncoupled

$$\begin{bmatrix} X & O & O \\ X & X & O \\ X & X & X \end{bmatrix}$$

Decoupled

$$\begin{bmatrix} X & X & X \\ X & X & X \\ X & X & X \end{bmatrix}$$

Coupled

A design must be decomposed deeply enough to have the mathematical relationship between FRs and DPs and to select parameters or dimensions as key variables to calculate information content as a quantitative value for a given FR_i which is defined as

$$I_i = \log_2 \frac{1}{p_i} = -\log_2 p_i,$$

, where p_i is probability of success. The total information content can be formulated as below based on the definition of information content for an FR. For the uncoupled design,

$$I_{sys} = -\log_2 p_{\{m\}} = \sum_{i=1}^m \log_2 p_i, \text{ where } p_{\{m\}} = \prod_{i=1}^m p_i$$

and for the decoupled design,

$$I_{sys} = -\log_2 p_{\{m\}} = \sum_{i=1}^m \log_2 p_{i|\{j\}}, \text{ where } p_{\{m\}} = \prod_{i=1}^m p_{i|\{j\}} \text{ for } \{j\} = \{1, 2, \dots, i-1\}$$

Here, $p_{\{m\}}$ is a joint probability that all m FRs are satisfied, and $p_{i|\{j\}}$ is the conditional probability of satisfying FR $_i$ given that all other relevant (correlated) $\{FR_j\}_{j=1,2,\dots,i-1}$ are also satisfied. The probability of success is defined as the portion of common range in which system range (SR) is overlapped inside design range to the design range (DR). The larger the portion of common range is, the higher the probability of success is.

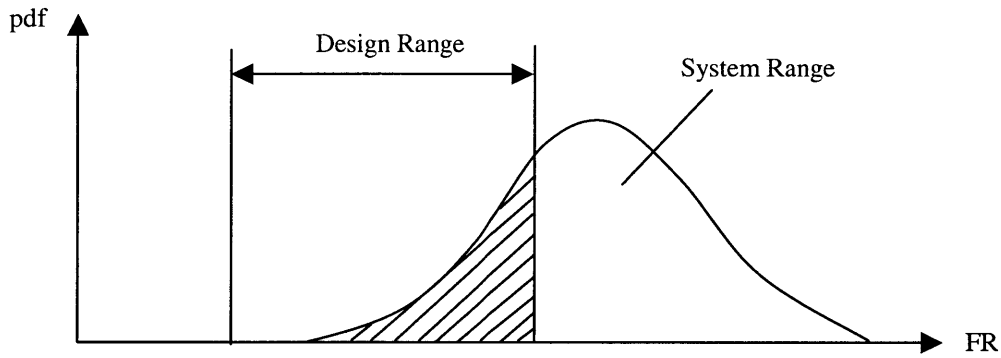


Figure 2. Design Range (DR) and System Range (SR)

Chapter 3

Design Process

3.1 V-Model

In artificial intelligence, top-down or bottom-up approach has been used to represent the human brain model. Top-down approach is represented as a process to decompose a big problem or a big structure into small pieces from top to bottom. The decomposed concept or knowledge is pre-programmed and stored into a large knowledge base. Then, inference techniques based on deduction, induction and/or abduction are used to infer relevant concept or knowledge from given queries to solve a given problem. Natural language processing [18] or CYC project [8] are the representative examples of the top-down approach. In contrast to the top-down approach, bottom-up approach begins with a relatively small number of physical building blocks. It interfaces the physical building blocks in the physical space, and finds out solutions by simulating their various combinations as alternatives to satisfy a certain goal. This approach has been used for robotics or artificial life [19, 20] using neural networks or genetic algorithms.

As a design process, the top-down approach requires a large knowledge base and extracts corresponding knowledge if the required design knowledge exists inside the knowledge base, but it has a small degree of freedom for creative design, because the traditional top-down approach lacks the re-composition process of combining extracted pieces of various design concept or knowledge. The bottom-up approach has more degrees of freedom to generate physical shapes or assemblies in the physical space that human designer

have not considered. However, the combination of the building blocks is sometimes too large to determine the best solution. Thus, the bottom-up process has been applied to small ad hoc problems or confined to a limited number of physical building blocks due to the computational complexity. Also, physical assembly without explicit description of FR must have a limited utility.

A V-model, adapted from Do and Suh [21], is proposed as a thinking process for computer-aided design of geometric topologies and shapes, which is based on the idea that design cannot proceed without zigzagging between the functional domain and the physical domain. It combines the advantages of the top-down and the bottom-up approaches and consists of three sub-processes as shown in Figure 3: top-down decomposition process, mapping process, and bottom-up integration process. Top-down decomposition process is the zigzagging decomposition process used in axiomatic design process. The decomposition process conceptually divides a big, complex problem into solvable small pieces and finds design solutions for the divided small problems. It produces language descriptions of decomposed FRs and DPs. A DP is a language description of a proposed solution to satisfy the corresponding FR, and plays a role as a key design variable as a part of the whole design solution.

Mapping process is a process to create geometric entities to the leaf level DPs produced by the top-down decomposition process. The geometric entities mapped to the leaf level DPs are key geometric objects that satisfy the corresponding leaf level FRs. They must be interfaced with each other and be integrated into a complete physical model in the physical domain to satisfy higher-level FRs through the bottom-up integration process. There are three steps in the bottom-up integration process. First step is to establish interfaces between the physical entities including geometric entities created during the mapping process. Second step is to construct topologies by integrating related geometric entities into complete solids. The construction of the topologies is determined by locating the geometric entities and connecting them with boundaries or connecting volumes in the physical space.

In this thesis, topology is defined as the number of faces. Thus, either the different number of faces are considered as different topologies of two geometric artifacts in this sense. Third step is the determination of the final shapes. The determination of the final shape is performed by changing locations, orientations, and/or dimensions of the geometric entities to satisfy all the FRs. The constructed topology constrains the change of the geometric entities and therefore affects to the satisfaction of FRs. The design matrix, shown in chapter 2, is used to visually relate each geometric entity, mapped to each DP, to each FR to check functional couplings. An illustrative example about the use of the V-model and the design matrix for geometry design will follow in section 4.

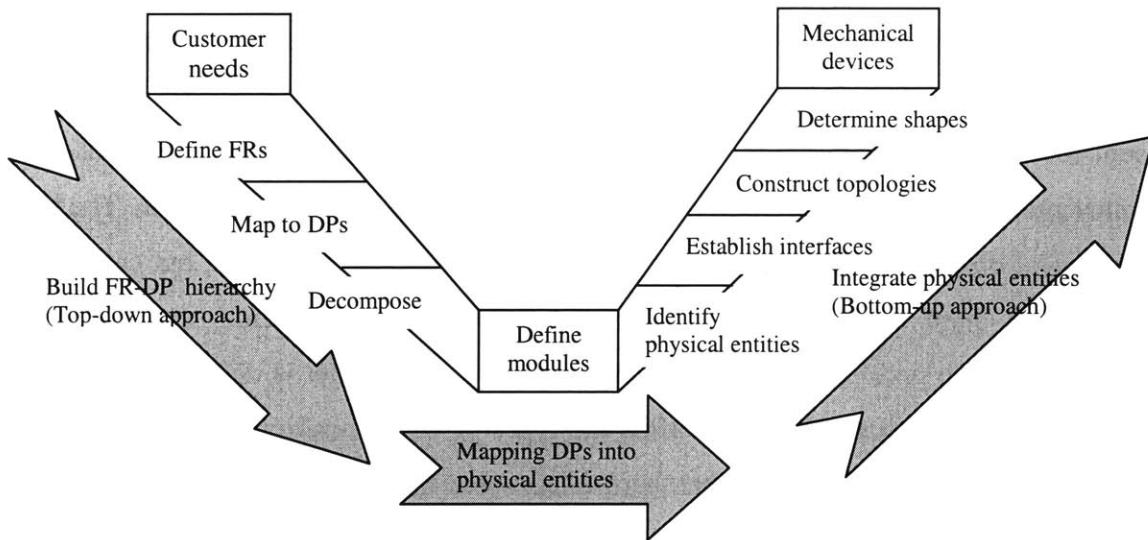


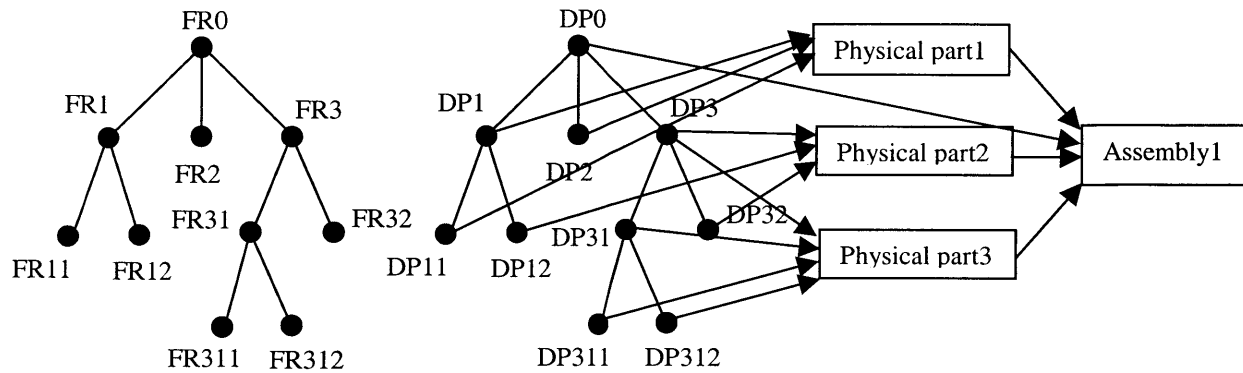
Figure 3. Axiomatic thinking process for designing physical products

The V-model design process finally produces an FR tree, a DP tree, and a final shape as design solutions for a design problem as shown in Figure 4. From the nature of the zigzagging decomposition process between FRs and DPs, all the DPs in the same level are sub-sets or elements of their parent DP in one-level higher, because the corresponding FRs come from the parent DP. For example, DP2 is a child of DP0, and DP1 and DP3 are sub-sets of DP0. DP11 and DP12 are elements of DP1 and DP31 is a sub-set of DP3. DP311 and

DP312 are elements of DP31. In this sense, DPs become more detailed as the decomposition is performed further. We propose zigzagging decomposition as an important design process, because lower level FRs can not be decomposed unless a concept of the higher level DP has been already determined. In Figure 4, FR1, FR2, and FR3 can not be determined, unless a concept on DP0 has been determined. Thus, FR1, FR2, and FR3 are functional requirements for DP0.

Most design theories using decomposition strategies looked over this fact or did not explicitly explain it. Taura and Yoshikawa [22] tried to mathematically represent a design and a design process for an intelligent CAD system based on General Design Theory proposed by Yoshikawa. [23] They proposed paradigm model and function concept set. The paradigm model is similar to the concept of zigzagging decomposition, but their mathematical model is not correct, because functional requirements and design solutions are mixed as functional concept sets. Thus, the intersection of some parts of physical objects in the lower level can not be represented as a higher level physical object to satisfy the higher level function. The higher level physical object must be synthesized from lower level physical entities, but cannot be an intersection of them. Umeda et al. [24] developed the function-behavior-state modeler and decomposed design problems between functions and behaviors, that is defined as ‘sequential state transition along time.’ They classified the types of decomposition into causal and task decomposition. However, we argue that there exists some cases in which behavior cannot be a design solution. Gero et al. [25] defined behavior with ‘a role as a link between function and structure’. Their definition of behavior was how the structure of an artifact achieves its functions. This definition is the same as that of DP in axiomatic design. They claimed that behavior support explicit reasoning between functions and structures. In contrast to those approaches to design process, we did not try to formalize the mapping between functions and geometric entities using behavior, but just used a direct mapping between them, because it is a more general representation of design problems. It is not easy to explicitly explain the mapping between functions and physical solutions in many design problems. Also, the trial to represent the mapping is a very laborious work for designers. Therefore, our design process is

more simple and general. Also, it can be applied to any design problem for designing a new artifact from scratch.



$DP1 \subset DP0, DP2 \in DP0, DP3 \subset DP0$

$DP11 \in DP1, DP12 \in DP1$

$DP31 \subset DP3, DP32 \in DP3, DP311 \in DP31, DP312 \in DP31$

Figure 4. Fundamental structure of decomposed FRs, DPs, and geometric entities

3.2 Illustrative Examples of V-Model

Two illustrative examples show how the proposed design process is applied to create design solutions as a thinking process for designing geometric artifacts. First example is a cam-pawl mechanism design to satisfy “prevent the vehicle from moving on its own.” Second example is the design of a hanger to satisfy “hang a load on the wall with a certain stiffness.” We have explicitly represented a thinking process to create two design solutions for these two examples based on the V-model.

3.2.1 Pawl and Cam Mechanism Design

This example shows how the proposed design process is applied to a design of a simple cam and pawl mechanism used for a parking mode in automatic transmission of automobiles.

Step 1: FR-DP decomposition

In this step, the zigzagging decomposition process produces FR-DP hierarchies. FRs are defined as “what we want to achieve” in functional domain and DPs are defined as “how we want to achieve it” in physical domain. Language description is used to explicitly represent the meaning of FRs and DPs. In this example, FR1, FR2, FR3, and FR4 are language descriptions, decomposed from DP0 for DP0 to function correctly satisfying FR0. The real physical shapes of solid parts and their assemblies might be in the designer’s mind, but have not been explicitly represented in this step. The followings are the decomposed FRs and DPs from top to bottom.

FR0 = Prevent an attended parked vehicle from moving on its own

FR1 = Engage the pawl to the engaged position

FR11 = Push the cam

FR12 = Push up the pawl to the engaged position

FR2 = Keep the pawl in the engaged position

FR3 = Disengage the pawl from the engaged position

FR31 = Pull out the cam

FR32 = Pull down the pawl from its engaged position

FR4 = Carry the weight of the vehicle in the engaged position

FR41 = Carry the force to the pin

FR42 = Endure the force at the pin

DP0 = Assembly of cam and pawl mechanism

DP1 = Engagement mechanism

DP11 = Pushing force

DP12 = Tapered section of cam

DP2 = Flat surface of cam

DP3 = Disengagement mechanism

DP31 = Pulling force

DP32 = Tension spring

DP4 = Force carrying mechanism

DP41 = Vertical surface of the pawl

DP42 = Pin

Step 2: Mapping DPs into geometric entities at the leaf-level

One of the most important features for axiomatic design theory is the design matrix. The design matrix is a relational basis for controlling DPs to satisfy FRs based on the relationship between FRs, DPs, and geometric entities. The couplings between FRs of each design alternative are checked by the design matrix. In this step, only diagonal elements are checked as shown in the Figure 5.

			DP0						
					DP2				
			DP11	DP12		DP31	DP32	DP41	DP42
FR0	FR1	FR11	X						
		FR12		X					
	FR2				X				
	FR3	FR31				X			
		FR32					X		
	FR4	FR41						X	
		FR42							X

Figure 5. Full design matrix of pawl-cam design

A designer creates key geometric entities using the geometry sketch tool and links them to corresponding language descriptions of the DPs at the leaf level. Each geometric entity is defined by its location, orientation, and shape parameters in the physical space. DP-geometry database stores information on the definitions of geometric entities as briefly explained in section 3.2. Table 1 shows the created geometric entities at the leaf level.



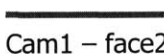


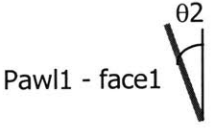

DP11		DP12	
DP2			
DP31		DP32	
DP41		DP42	

Table 1. Geometric entities mapped from leaf level DPs of the pawl-cam design

Step 3: Establish interfaces

All interfaces between geometric entities must be explicitly defined and represented during this step. A process of defining the interfaces using the geometry sketch tool makes the designer to represent all the geometric entities and their interfaces explicitly in a computational form. In this example, information on the pawl, the designer considered in the top-down decomposition process, but not shown as any DP, is explicitly represented and visualized during this step. Table 2 shows all the interfaces between geometric objects that include all the leaf level DPs. As shown in the table, one level higher DPs are constructed in this step.

DP1	DP2
DP3	DP4

Table 2. Defined interfaces and corresponding DPs of the pawl-cam design

Step 4: Construct topologies

The created and interfaced geometric entities through the previous steps are integrated into certain topologies in this “construct topologies” step. Three procedures are needed to complete this step:

- (1) Collecting related geometric entities that consist of complete solid parts based on their names or language descriptions
- (2) Locating the geometric entities in the physical space

- (3) Skinning by making boundaries between geometric entities and then filling materials inside the closed boundaries

Topology construction is crucial for the performance of the final product and is closely related to creativity. Generally, designers use sketches to create various topologies. However, topology construction procedure in the physical space is a very tedious even with a small number of geometric entities. Therefore, a computer support is needed for constructing various topologies with only key geometric entities. Figure 6 shows possible topologies for DP0 integrated from leaf level DPs. The three alternatives are generated by positioning the same geometric entities in the different locations. Alternative 1 and 3 have the same topologies based on the definition of the topology, because each part has the same number of faces. However, the topology of pawl in alternative 2 is different from others.

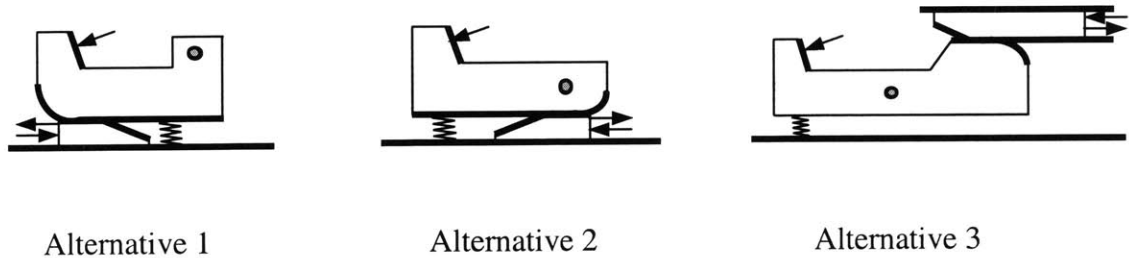


Figure 6. Possible alternatives based on topology construction of the pawl-cam design

Step 5: Determine shapes to satisfy given FRs

The best shape for each alternative design generated through the step 4 is determined by controlling DPs to satisfy the corresponding FRs. The FRs, described by language in step 1, can be represented by using mathematical equations to calculate system ranges, which satisfy given design ranges as shown in several engineering design problems of [4, 5]. In this example, six mathematical equations have been made for alternative 1 to calculate system ranges. Figure 7 shows the links of FRs to the mathematical equations for each module (“SR” stands for “System Range”), and the links of DPs to geometric entities created in the physical space on the computer screen. SR1 is the difference between the pushing force (DP11) and

the friction force generated from an interface between the pawl and the tapered section of cam (DP12). Thus, SR1 should be greater than zero to engage the pawl to the engaged position. SR2 is a safety factor to keep the cam at the engaged position from any disturbances. SR31 is the difference between the pulling force (DP31) and the friction force generated from an interface between the pawl and the flat surface of the cam (DP2). SR32 is a sum of vertical forces in downward direction to disengage the pawl from the engaged position after pulling out the cam. SR41 is a force applied to the pin carried by the vertical surface of the pawl (DP41) from the given force generated by the weight of the vehicle. SR42 is a maximum stress produced in the pin (DP42). The system ranges are calculated based on dimensions shown in Figure 8 based on assuming quasi-static equilibrium. Once design ranges are given, each geometric entity can be controlled to move the system range inside the design range. As shown in Figure 7, SR32 and SR41 are coupled. Thus, change of either DP32 or DP41 will affect both SR32 and SR41, requiring iterations to satisfy given design ranges for FR32 and FR41. The procedure to satisfy all the given design ranges determines the best shape to satisfy the FRs of the design task. The determined shapes of the product on the sketch tool are generated in solid models for further analysis in detailed design stage. Figure 7 shows one possible shape for alternative 1 by given arbitrary design ranges.

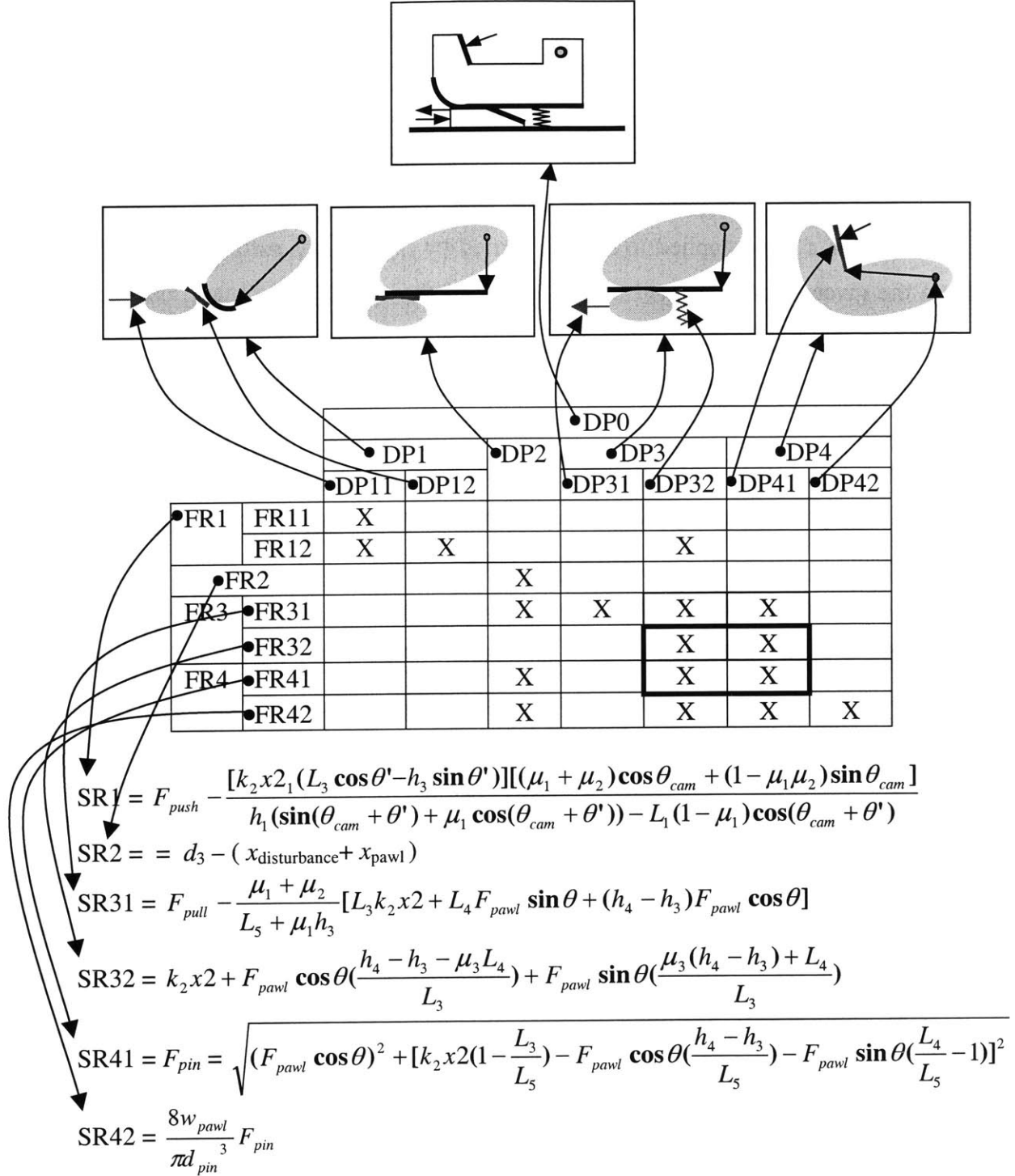


Figure 7. Links of FRs to geometric entities based on the design matrix for alternative 1

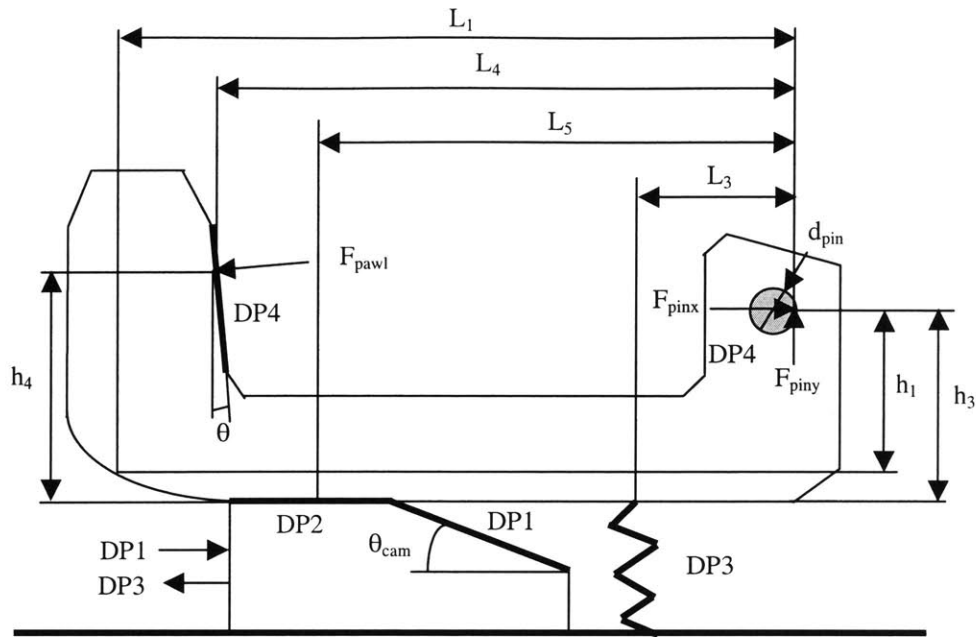


Figure 8. DPs and dimensions of alternative 1

3.2.2 Hanger Design

This example is much simpler than the pawl and cam design problem, but explicitly shows the concept of independence axiom. The design problem is to design a hanger to hang a load on the wall.

Step 1: FR-DP decomposition

FR0 = hang a load on the wall with a certain stiffness

FR1 = hang a load

FR2 = increase stiffness

DP0 = hanger

DP1 = L-type beam

DP2 = rib

Step 2: Mapping DPs into geometric entities at the leaf-level

In this step, a full design matrix is constructed with only diagonal elements. The L-type beam is mapped to DP1 and three types of ribs are mapped to DP2 as candidate design solutions to satisfy FR1 and FR2. Figure 9 shows the diagonal elements between FRs and DPs and Table 3 shows the corresponding geometric entities for DP1 and DP2.

		DP0	
		DP1	DP2
FR0	FR1	X	
	FR2		X

Figure 9. Full design matrix of the hanger design

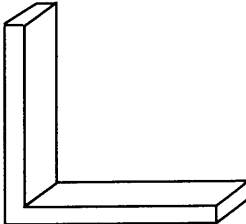
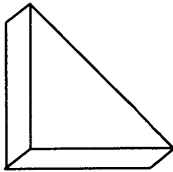
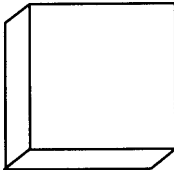
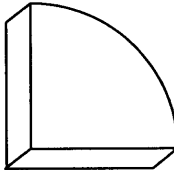
DP1	DP2				
 L-type beam	 Triangular rib	OR	 Rectangular rib	OR	 Circular rib

Table 3. Geometric entities mapped from leaf level DPs of the hanger design

Step 3: Establish interfaces

Interface has been defined in this step. In this example, two concave faces of the L-type beam and two convex faces of each rib, both of which are gray and have 90 degree, are interfaced with each other. Table 4 shows an interface between the L-type beam and the triangular rib. The interface is a rigid attachment of the two geometric entities. Other two ribs have the same interface type.

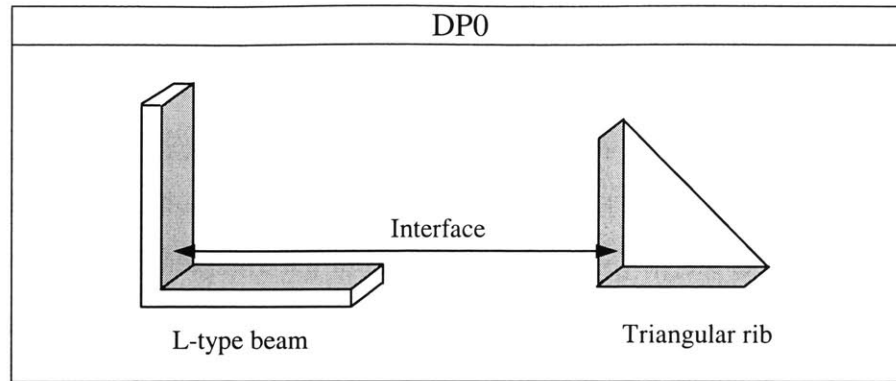


Table 4. Defined interfaces and corresponding DPs of the hanger design

Step 4: Construct topologies

Interface between the L-type beam and each of the three ribs generates three candidate solutions with different topologies as shown in Table 5. As we defined in section 3.1, the different face types mean the different topologies. Thus, three candidate solutions are considered as they have different topologies. The three alternatives are created by the combination of different geometric entities differently from the alternatives generated by different locations of geometric entities in the pawl-cam design.

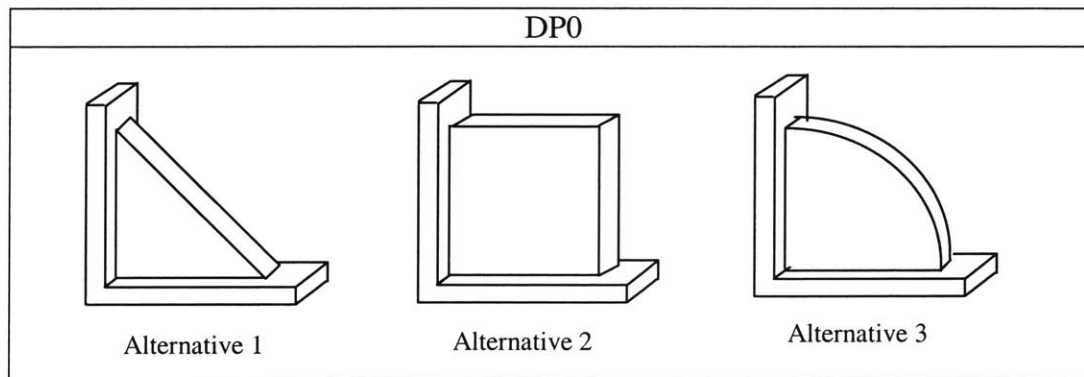
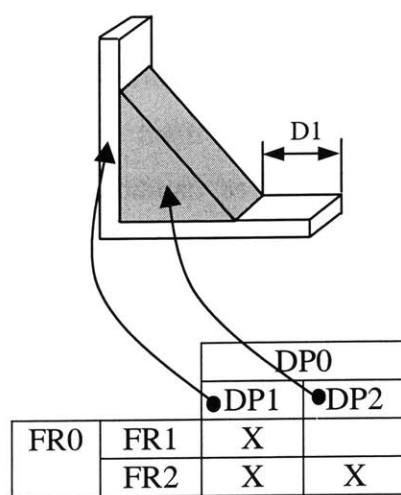


Table 5. Design alternatives of the hanger design

Step 5: Determine shapes to satisfy given FRs

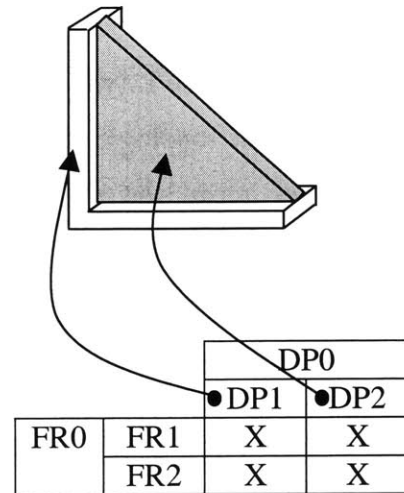
The FRs and geometric entities are visually related with each other through the design matrix as shown in Figure 10. The candidate shape 1 has a decoupled design matrix, because the L-type beam affects the stiffness of the hanger, but the triangular rib does not affect FR1 to hang a load on the beam. The SR1 is D1 set to 15 cm to keep the space to hang a load on the beam in the candidate shape 1. However, the candidate shape 2 has a coupled design matrix in which DP2 affects FR1. In this case, both FRs cannot be satisfied. SR2 for both candidate shapes are calculated by using finite element analysis.



$$SR1 = D1 = 15 \text{ cm}$$

SR2 = stiffness

(a) Decoupled candidate shape 1



$$SR1 = D1 = 0 \text{ cm}$$

SR2 = stiffness

(b) Coupled candidate shape 2

Figure 10. Links of FRs to geometric entities based on the design matrix for alternative

Chapter 4

Database

4.1 System Architecture

Figure 11 shows the system architecture and the flow chart. The flow chart shows how the proposed method can help designers, who are designing geometric artifacts within the V-model design process, to search geometric solutions as an assembled candidate shapes. The existing design knowledge must be collected and stored in the database before the designers use the database. The top two blocks on the flow chart show this knowledge engineering process. It will be explained in detail in this section. The flow chart is modified from that of Thinking Design Machine proposed by Suh and Sekimoto. [26] The proposed CAD system consists of knowledge engineering tool, databases, CAD engine, and graphical user interface. The knowledge engineering tool supports knowledge engineers or axiomatic designers to transform existing designs and CAD models into properly formatted design knowledge and to store it into the database. The necessary information on geometry can be easily constructed by using current CAD packages, because only information on mating faces of FGFs is needed. This is relatively simple comparing to most approaches using complex geometric information on the shape of the artifacts. The CAD engine searches a plausible set of DPs and FGFs by language matching between input FR and stored FRs. The similarity between FR statements is calculated using a value defined in `FR_metric` between 0.0 and 1.0. It also simulates assembly of FGFs and calculates the interface-ability using a value defined in `INTERFACE_metric` between 0.0 and 1.0. The generated candidates are evaluated by constructing design matrices as explained in chapter 3.

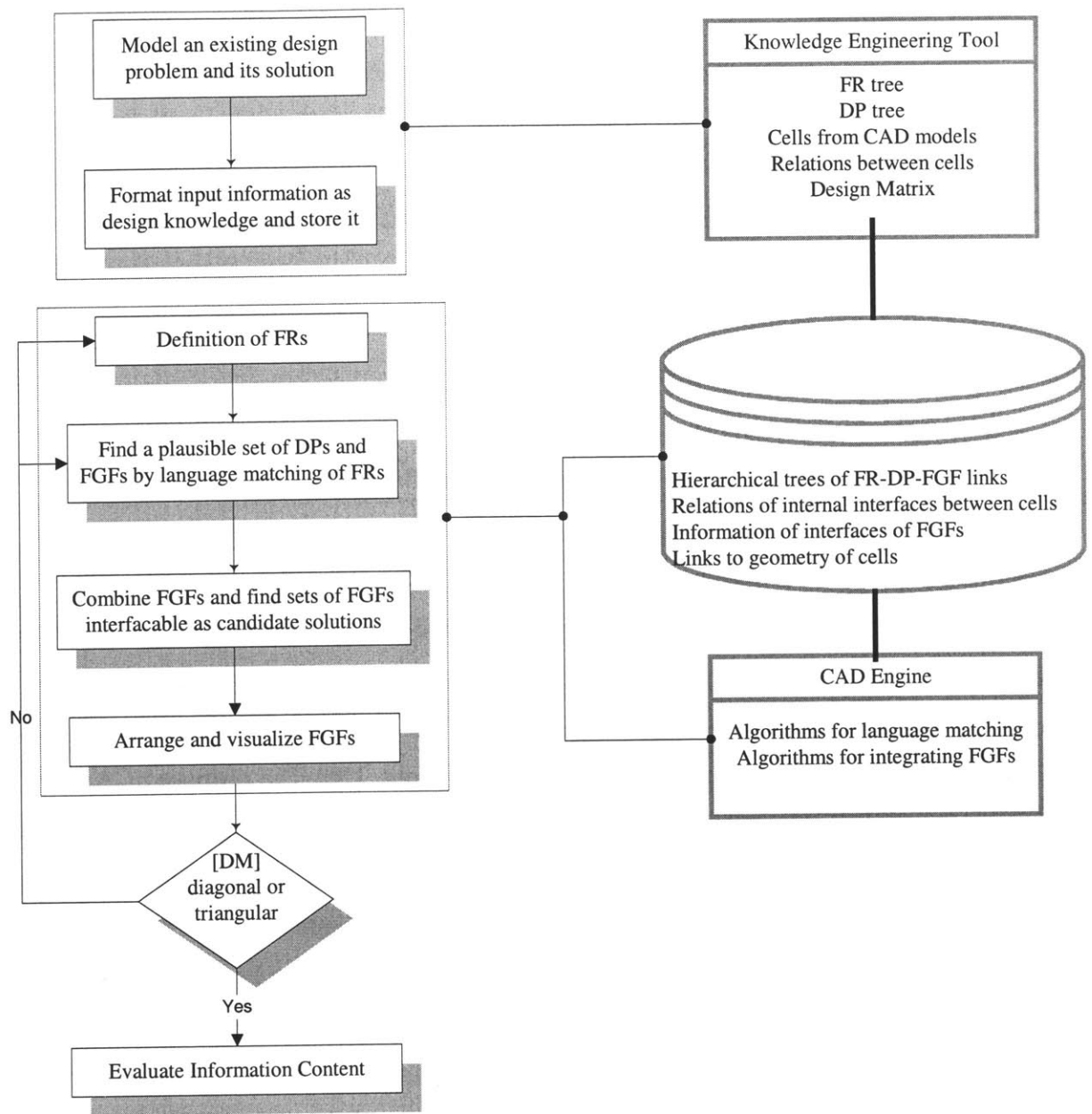


Figure 11. System architecture and flow chart

4.2 Data Structure

The V-model design process generates two types of design knowledge; one is knowledge about how FRs, DPs, and geometric entities are related with each other, and the other is knowledge about how FRs and DPs are decomposed in depth. Figure 12 shows the design databases, which store both types of design knowledge. It consists of databases expressed as:

```
fr-decomposition("A", [FR0(FR1(FR1_1, FR1_2),  
                             FR2  
                             FR3(FR3_1(FR3_1_1, FR3_1_2), FR3_2))])  
  
dp-decomposition("A", [DP0(DP1(DP1_1, DP1_2),  
                             DP2  
                             DP3(DP3_1(DP3_1_1, DP3_1_2), DP3_2))])  
  
dp-geometry("DP1_A", "FGF1_A")
```

Figure 12. Fundamental structure of database

It is used to search for proper candidate DPs and FGFs for a given FR. The fr-decomposition database stores information of hierarchies of decomposed FRs and the dp-decomposition database stores information of hierarchies of decomposed DPs. A character, "A" is an index to the corresponding design case. The dp-geometry database stores information of FGFs mapped to the corresponding solid cells. The fr-decomposition database and the dp-decomposition database can be constructed through the V-model design process. However, additional operations are needed to generate information on FGFs from CAD models for the reuse of them.

4.3 Functional Geometric Feature (FGF)

The term “functional geometric feature (FGF)” is used to represent a set of geometric entities mapped to an FR through a DP, because the mapping between geometric entities and a DP ultimately cause a mapping between the geometric entities and an FR. Actually, a DP can be mapped to a set of any geometric entities such as vertices, edges, faces, solids, and/or assemblies of CAD models in axiomatic design theory. Thus, shape decomposition is needed to separate the corresponding geometric entities and to keep information on the separated geometric entities.

The shape decomposition starts from selecting “a set of faces,” which represents a DP. Any geometric entity can be represented by a set of faces. For example, if a vertex or an edge is the element of FGF, the faces including the vertex or the edge can be selected to decompose the shape. If a solid or an assembly is the element of FGF, a set of faces, which composes of the solid or the assembly, can be selected. Based on the visualized CAD models, the knowledge engineer selects a set of faces including the corresponding geometric entities and then decomposes the CAD models into solid cells including the set of faces for each FGF. Two kinds of the shape decomposition operations have been considered; cut and subtraction. The operation, “cut,” cuts protruded solid parts from the base solid and the operation, “subtraction,” subtract a volume, which is defined by a set of faces, from the whole solid.

Our hypothesis on FGFs is

“Functional geometric features (FGFs) can be topologically and geometrically separated from a whole solid body or an assembly.”¹

This hypothesis must be true, if shapes are created through the V-model design process so that geometric entities such as vertices, edges, faces, etc., can be mapped to each FR. Even though this hypothesis is conceptually true, there exist more than one way to separate the geometric

¹ This is consistent with the Independence Axiom and the FR/DP mapping of axiomatic design.

entities in practice. This is inevitable, because an FGF is not defined by geometric characteristics but by FRs. In this thesis, three examples are considered to look into the way of separating the geometric entities. The designs and CAD models in the examples have been decomposed into solid cells considering reusability of the cells. The way of making the decomposed cells can be considered one design case. Thus, if the same design and CAD models are decomposed in different ways by different people, all the different decompositions can be stored in databases as different design cases. Some research related to knowledge bases approaches to finding common elements of many design cases and to store the elements as a generalized or common design knowledge in the database. [7] Others use concept of inheritance in object oriented representation to classify the design knowledge. [27] Even though those approaches are have some advantages (on search time or on inference of design knowledge), it is not easy to generalize design knowledge related to geometry. Our approach is not to generalizing all the specific design knowledge from all the design cases, but just to collecting design cases. The design knowledge collected is linked lists of FR-DP-FGF-INTERFACES and the corresponding decomposition trees.

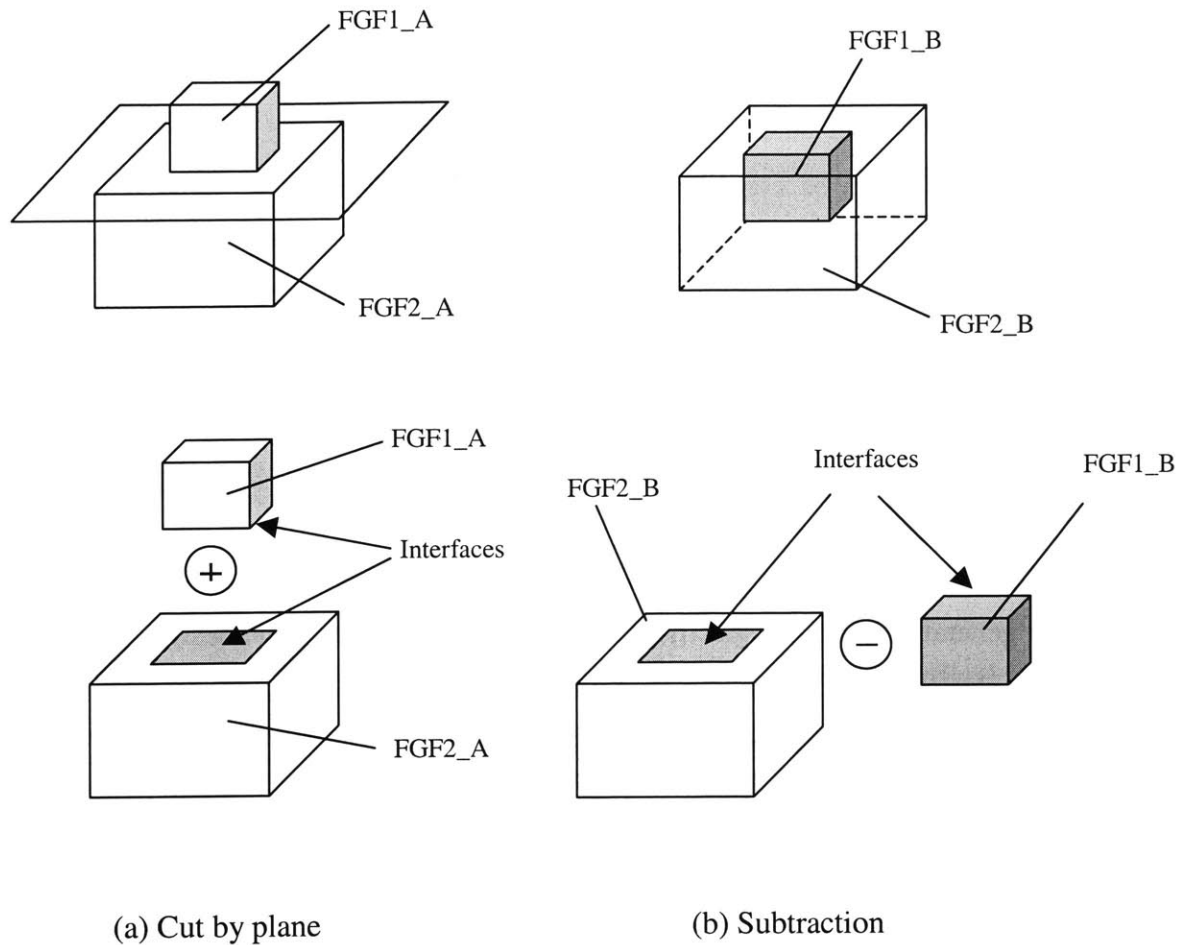


Figure 13. Operations: cut and subtraction

The guideline of separating correct FGFs satisfying a certain FR from the whole solid body for collecting a set of FGFs is as follows:

“To determine the correspondance between an FR and an FGF, the cells corresponding to the FGF can be removed from the whole solid body. If the resulting solid body can no longer satisfy the FR, the removed cells are the proper FGF for the FR.”

We can have Ω as the operator representing the entire decomposition procedure to obtain the cells. B represents the original solid bodies and C is a set of cells, which are decomposed by Ω . Then, we have:

$$\Omega(B) = C,$$

Each FGF is a set of cells. The cell is decomposed into either positive volume or negative volume as shown in Figure 13. The cells of a lower level FGF is a subset or elements of its parent FGF, because FGFs are defined by zigzagging decomposition between functional domain and physical domain as explained in the accompanying thesis. Each cell can be represented either type of representation, B-rep (boundary representation) [28] or face-edge graphs. Information on the geometries of the cell can be encapsulated, but only information on the interfaces of the cell is open for the computer algorithms to reuse the cell. Figure 14 shows face-edge graph representations of FGF1_A and FGF2_A shown in the Figure 13. All the faces and edges of each cell has been represented by a face-edge graph as shown in (b). The faces and edges can be encapsulated except mating faces as shown in (c). All the FGFs can be regenerated from the existing CAD models using current CAD packages. SolidWorks has been used in this thesis. Necessary information can be successfully extracted from STEP files shown in [29]. The attributes associated with a node in a face-edge graph are related to information on a face. For example, interface nodes, D and b, have the following attributes as mating faces.

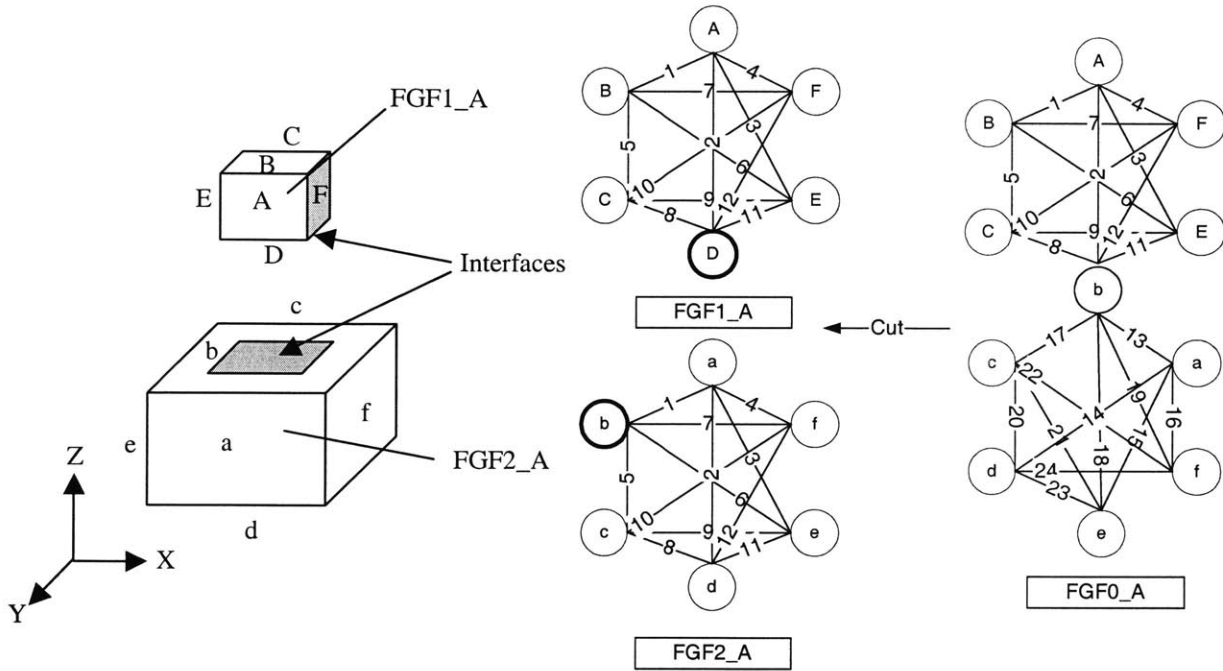
- Node_name D
- Cell_name cell_1
- Face_type planar
- Dir_normal (0, 0, -1)
- Interface_type rigid_attachment
- Mating_type against

- Node_name b
- Cell_name cell_2
- Face_type planar
- Dir_normal (0, 0, 1)
- Interface_type rigid_attachment
- Mating_type against

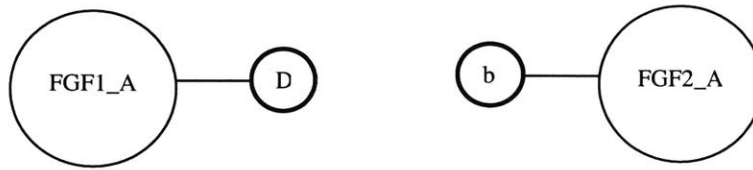
The attributes associated with an arc are related to information on an edge and on the faces connected to the edge. An arc 11 of FGF1_A has the following attributes.

- Edge_name 11
- Cell_name cell_1
- Node_1 D
- Node_2 E
- Edge_type linear
- Edge_geometry (x1, y1, z1), (x2, y2, z2)

A node can be differentiated from others by node_name and cell_name. Dir_normal is needed to attach this face to another face. An edge can be also differentiated from others by edge_name and cell_name. Node_1 and node_2 are the attached nodes to this edge in the same cell. Edge_geometry has information on the geometry of the edge such as vectors of two vertex points of the edge for 'linear' edge type. Bold lined nodes represent mating faces. A node D and b has been created by the operation, 'cut'.



(a) Geometric models (b) Graph representations of all the faces and edges of FGFs



(c) Graph representations with encapsulated geometry

Figure 14. Graph representations of FGFs

Interface types of mating faces can be classified as follows.

- Rigid attachment: two cells are merged into a solid. A face should be rigidly attached with another face, or a volumetric cell can be subtracted from a bigger volumetric cell.
- Assembly: two solid cells contact each other.
 - Static: two cells contact without relative motion between each other.

- Kinematic: two cells contact with relative motion between each other. This interface can be classified further into “slide”, “roll”, and “rotate”.

Two types of mating for assemblies between components such as ‘against,’ and ‘fits’ (or ‘tight-fits’) have been considered by Lee and Gossard,² and Lee and Andrews [31] also developed a way to infer the positions of the components from the given mating features. This approach gave the basics to calculate the locations of components by using mathematical equations and solving them. They considered only assemblies, but we consider rigid attachment between cells in this thesis. Thus, ‘against,’ and ‘fits’ can be extended as follows.




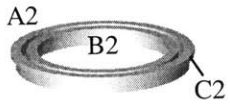
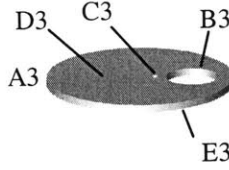

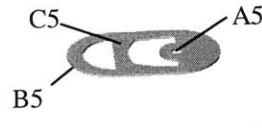
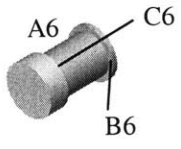
- Against:assembly: two planar faces are located on a plane with opposite normal vectors.
- Against_equal:rigid_attachment: all the boundaries of two planar faces are exactly matched together.
- Against_inside:rigid_attachment: a planar face is placed inside of the other planar face.
- Against_intersected:rigid_attachment: two planar faces are intersected on a plane.
- Fits:assembly: two centerlines of cylinder and hole are located on a line. This interface type is classified into ‘cylinder,’ or ‘hole’ for each mating face.
- Tight-fits: assembly: two centerlines of cylinder and hole are located on a line and two points on the cylindrical faces locates on a cylindrical face. This interface type is classified into ‘cylinder,’ or ‘hole’ for each mating face.

In this thesis, we grouped all kinds of rigid attachments into ‘against:rigid_attachment,’ because we are not considering shape deformation of the boundaries of the mating faces except some special cases. Deformation of the boundaries can be performed by post-processing since interface-ability has been checked only considering orientations and positions of the FGFs to satisfy given mating conditions. Thus, four types of interface, ‘against:rigid_attachment’ ‘against:assembly,’ ‘fits:assembly,’ and ‘tight-

² ‘Against’ applies between planar faces and ‘fits’ applies between a solid cylinder and a hole.[30]

'fits:assembly,' are mainly considered, but only two types of interface, 'against' and 'fits' are used for automatic assembled shape generation. These two types of interface can cover many cases for assemblies as mentioned in [30]. For the 'against,' normal vector of the planar face and one point on the face are needed for reusing the face for geometric interfaces. For the 'fits,' two points on the centerline of each cylindrical face are needed. Also, an operation, 'subtraction,' is not considered in this thesis.

Table 6 shows a simple practice of how to generate the FGFs for a beverage can. The generated FGFs are decomposed by simple disassemblies and cut operations in the V-model design process. Figure 15 shows the detailed data structure between FGFs and cells for the beverage can example. Each leaf level FGF has been linked to each cell. Each cell keeps information on its interfaces, i.e. mating faces, and encapsulated geometry. Each higher level FGF is composed of its lower level FGFs. This can be represented as a set of lower level FGFs. Because the lower level FGFs are linked to cells, each higher level FGF is consequently represented by a set of cells. FGF_0, FGF_1, FGF_2, or FGF_3 does not have all the geometric information of the corresponding shapes, but only keeps a set of links to the corresponding cells, a list of internally matched interfaces between the cells, and dangling interfaces of each FGF that have not been interfaced yet. The geometry of the FGFs and their dangling interfaces are shown in the Table 6. In the case, in which one mating face may be used for mating two other faces, two interfaces of the same mating face are generated and stored as design knowledge in the database. This gives complete information on how the interfaces have been used in the existing design case. For example, A6 (fits:assembly:cylinder), a mating face of cell 6, has been interfaced with two faces, C3 (fits:assembly:hole) of cell 3 and A5 (fits:assembly:hole) of cell 5.

FRs	DPs	FGFs	Dangling interfaces
FR0 = contain and carry beverage	DP0 = beverage can		{FGF_1, FGF_2, FGF_3}
FR1 = cover the body and provide pathway of beverage	DP1 = top		{FGF_1_1, FGF_1_2, FGF_1_3, FGF_1_4, FGF_1_5, FGF_1_6} C2(against:rigid_attachment)
FR1_1 = guide the flow of beverage	DP1_1 = cylindrical extrusion wall		{Cell_1} A1(against:rigid_attachment)
FR1_2 = store beverage	DP1_2 = circular pocket		{Cell_2} A2(against:rigid_attachment) B2(tight-fits:assembly:hole) C2(against:rigid_attachment)
FR1_3 = cover body of beverage can and flow beverage out	DP1_3 = circular cover with through hole		{Cell_3} A3(tight-fits:assembly:cylinder) B3(tight-fits:assembly:hole) C3(tight-fits:assembly:hole) D3(against:assembly) E3(against:assembly)
FR1_4 = enclose the hole before opening	DP1_4 = gate		{Cell_4} A4(tight-fits:assembly:cylinder)
FR1_5 = open the gate	DP1_5 = opening tab		{Cell_5} A5(tight-fits:assembly:hole) B5(against:assembly) C5(against:assembly)
FR1_6 = fasten opening tab on the cover	DP1_6 = fastener of opening tab to cover		{Cell_6} A6(tight-fits:assembly:cylinder) A6(tight-fits:assembly:cylinder) B6(against:assembly) C6(against:assembly)

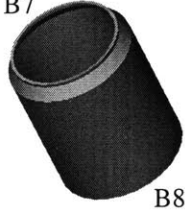



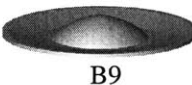

FR2 = provide volume to contain beverage with a certain stiffness	DP2 = cylindrical body		{FGF_2_1, FGF_2_2} B7(against:rigid_attachment) B8(against:rigid_attachment)
FR2_1 = reduce the material and increase stiffness of the body	DP2_1 = conical section of the body		{Cell_7} A7(against:rigid_attachment) B7(against:rigid_attachment)
FR2_2 = provide a volume to contain beverage	FR2_2 = hollow cylinder		{Cell_8} A8(against:rigid_attachment) B8(against:rigid_attachment)
FR3 = enclose the bottom with resistance to an impact	DP3 = bottom		{FGF_3_1, FGF_3_2} A9(against:rigid_attachment)
FR3_1 = withstand a moderate impact when the can is dropped	DP3_1 = curvature of the bottom		{Cell_9} A9(against:rigid_attachment) B9(against:rigid_attachment)
FR3_2 = stand the beverage can and stack on top	DP3_2 = cylindrical extrusion		{Cell_10} A10(against:rigid_attachment)

Table 6. FRs, DPs and the corresponding FGFs of beverage can











Dangling Interface	Internal Interface Match	FGF	Pointer	Cell	Interface	Geometry
Null	{FGF_1, FGF_2, FGF_3} (C2:B7) (B8:A9)	FGF_0		Cell_1	A1	
C2	{FGF_1_1, FGF_1_2, FGF_1_3, FGF_1_4, FGF_1_5, FGF_1_6} (A1:A2) (B2:A3) (B3:A4) (C3:A6) (D3:B5) (E3:B6) (A5:A6) (C5:C6)	FGF_1		Cell_2	A2, B2, C2	
A1	{Cell_1}	FGF_1_1		Cell_3	A3,B3,C3,D3,E3	
A2, B2, C2	{Cell_2}	FGF_1_2		Cell_4	A4	
A3,B3,C3,D3,E3	{Cell_3}	FGF_1_3		Cell_5	A5, B5, C5	
A4	{Cell_4}	FGF_1_4		Cell_6	A6, A6, B6, C6	
A5, B5, C5	{Cell_5}	FGF_1_5		Cell_7	A7, B7	
A6, A6, B6, C6	{Cell_6}	FGF_1_6		Cell_8	A8, B8	
B7, B8	{FGF_2_1, FGF_2_2} (A7:A8)	FGF_2		Cell_9	A9, B9	
A7, B7	{Cell_7}	FGF_2_1		Cell_10	A10	
A8, B8	{Cell_8}	FGF_2_2				
A9	{FGF_3_1, FGF_3_2} (B9:A10)	FGF_3				
A9, B9	{Cell_9}	FGF_3_1				
A10	{Cell_10}	FGF_3_2				

Figure 15. Data structure between FGFs and cells for beverage can

The following Table 7 and Figure 16 show generated data from the cam and pawl mechanism and Table 8 and Figure 17 show the hanger design example described in the chapter 3.

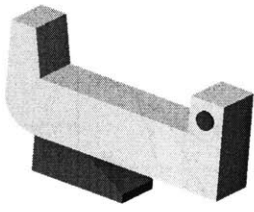
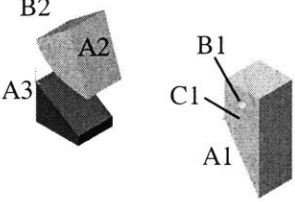
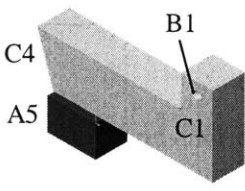
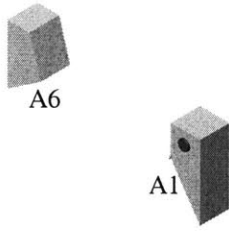
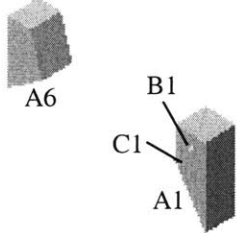

FRs	DPs	FGFs	Dangling interfaces
FR0 = prevent an attended parked vehicle from moving on its own	DP0 = assembly of cam and pawl mechanism		{FGF_1, FGF_2, FGF_4}
FR1 = engage the pawl to the engaged position	DP1 = engagement mechanism		{Cell_1, Cell_2, Cell_3} A1(against:rigid_attachment) B1(tight-fits:assembly:hole) C1(against:assembly) A2(against:rigid_attachment) B2(against:rigid_attachment) A3(against:rigid_attachment)
FR2 = keep the pawl in the engaged position	DP2 = flat surface of cam		{Cell_1, Cell_4, Cell_5} B1(tight-fits:assembly) C1(against:assembly) C4(against:rigid_attachment) A5(against:rigid_attachment)
FR4 = carry the weight of the vehicle in the engaged position	DP4 = force carrying mechanism		{FGF_4_1, FGF_4_2} A1(against:rigid_attachment) A6(against:rigid_attachment)
FR4_1 = carry the force to the pin	DP4_1 = vertical surface of the pawl		{Cell_1, Cell_6} A1(against:rigid_attachment) B1(tight-fits:assembly) C1(against:assembly) A6(against:rigid_attachment)
FR4_2 = endure the force at the pin	DP4_2 = pin		{Cell_7} A7(tight-fits:assembly:cylinder) B7(against:assembly)

Table 7. FRs, DPs and the corresponding FGFs of cam and pawl mechanism

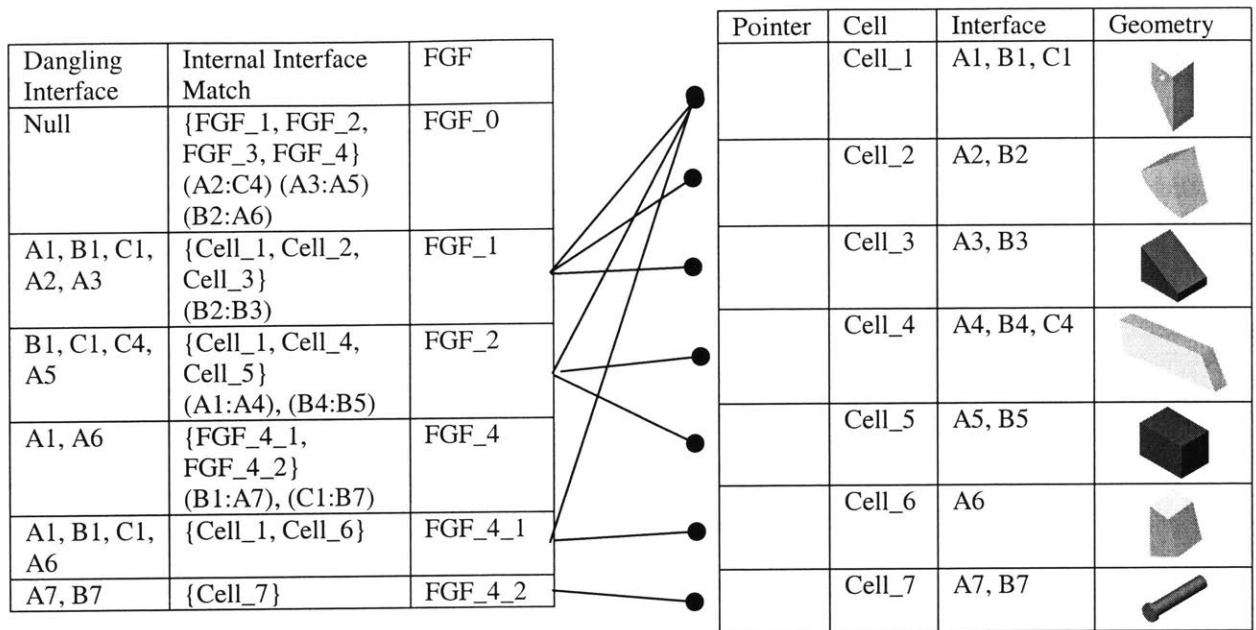


Figure 16. Data structure between FGFs and cells of cam and pawl mechanism

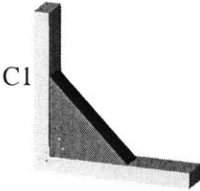
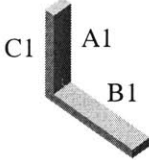
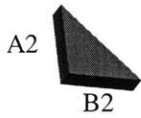
FRs	DPs	FGFs	Dangling interfaces
FR0 = hang a load on the wall with a certain stiffness	DP0 = hanger		{FGF_1, FGF_2} C1(against:rigid_attachment)
FR1 = hang a load	DP1 = L-beam		{Cell_1} A1(against:rigid_attachment) B1(against:rigid_attachment) C1(against:rigid_attachment)
FR2 = increase stiffness	DP2 = triangular rib		{Cell_2} A2(against:rigid_attachment) B2(against:rigid_attachment)

Table 8. FRs, DPs and the corresponding FGFs of hanger design



Dangling Interface	Internal Interface Match	FGF	Pointer	Cell	Interface	Geometry
C1	{FGF_1, FGF_2} (A1:A2), (B1:B2)	FGF_0	●	Cell_1	A1, B1, C1	
A1, B1, C1	{Cell_1}	FGF_1	●	Cell_2	A2, B2	
A2, B2	{Cell_2}	FGF_2				

Figure 17. Data structure between FGFs and cells of hanger design

Chapter 5

Generation of Design Solutions

The first step of generation of design solutions is language matching between a query FR and saved FRs. Once there exist any candidate FGFs to satisfy the query FR, the interface-ability between the FGFs should be checked first and then those FGFs should be combined and integrated as candidate shapes. These steps reduce a design solution space gradually and finally can suggest alternatives for a design problem. Because the candidate design solutions has been filtered through the language matching first through the hierarchical tree, the computational complexity can be reduced to combine the FGFs to alternative shapes larger than that of bottom-up approach.

5.1 Language Matching of FRs

The search of design solutions is basically performed by text analysis of FRs. Once a designer input a query FR, a computer algorithm searches similar FRs to the query FR in the fr-database. If the language description of the query FR has similarity to an FR1 of hierarchy “A” and an FR2 of hierarchy “B” already stored in the case base, “DP1_A” and “DP2_B” can be thought of as different design solutions to satisfy the query FR. For one FR, many DPs can be found out as candidate solutions through this search process in the databases. The mapping between the query FR (FR0) and many candidate DPs can be expressed as

fr-dp(“FR0”, [“DP1_A”, “DP2_B”,]).

Each DP related to geometric entities is also linked to each FGF. Each FGF is defined by a set of nodes in the face-edge graph that represent a set of faces in the solid model. Thus, the representation of the mapping between FR0 and many candidate solutions related to geometric entities can be expressed as

$$\text{fr-dp-geometry}(\text{"FR0"}, [\{\text{"DP1_A"}, \text{"FGF1_A"}\}, \{\text{"DP2_B"}, \text{"FGF2_B"}\}, \dots]).$$

One difference in the structure of the case base proposed in this thesis from the database of the Thinking Design Machine is a tree structure of the knowledge base and the hierarchical information flow. This hierarchical information flow is more compact to store design knowledge, because all the lower level branches from a certain FR can be searched and extracted.

Search of the proper DPs from given FRs is performed by a statistical approach. The statistical approach measures the distance between an FR already stored in the case base and a query FR. The distance is calculated by the frequency of appearing words and the sequence of words. A query FR and a given saved FR are decomposed into a set of words and pairs of words. For example, “engage the pawl to the engaged position,” is decomposed into two sets as

Word set = {engage, pawl, to, engaged, position}

Word pair set = {(engage, pawl), (pawl, to), (to, engaged), (engaged, position)}.

Here, ‘the,’ ‘a,’ ‘and,’ ‘or,’ and pronouns are trimmed, because the word is not closely related to the meaning of the sentence. The word set is used to check all the words of the query FR are included in the given saved FR. The word pair set is used to check the sequence of the matched words. When the words are compared, synonyms of a certain English word are checked by WordNet [32]. The metric is

$$FR_metric = \frac{2 \times \# \text{ of matched words} + \# \text{ of matched word pairs of saved FR}}{\# \text{ of total words} + \# \text{ of matched word pairs of input FR}} \quad (a)$$

Once two sentences for a query FR and a given saved FR are exactly the same, this metric gives 100%. If not, it scales the difference by the mathematical equation defined for the metric. Synonym checking reduces errors of describing a meaning of a word using synonyms.

5.2 Assembled Shape Generation

Integration of various FGFs to a complete shape is one of the key factors for geometry design with functions, actually with FRs. To integrate FGFs, combining the candidate FGFs and checking of geometric interface-ability between them have to be done. The combination of FGFs is performed by genetic algorithm (GA) and geometric interface-ability is checked by an algorithm simulating assembling operations to integrate them in the physical space. The following attributes are checked to quantify the interface-ability between FGFs. Firstly, interface type (IT) is checked by graph matching of interfacing nodes of FGFs. Graph matching is known as NP-complete problem. [33] Secondly, relative angles between faces is measured. Thirdly, the possibility of positioning mating faces.

- Interface type (IT): against:assembly, against:rigid_attachment, fits:assembly
- Angles between faces (AF)
- Positions (Pos)

5.2.1 Combinations of FGFs

Three sets of candidate FGFs to satisfy FR1, FR2, and FR3 can be supposed as

```
{
fr-dp-geometry ("FR1", [{ "DP1_A", "FGF1_A" }, { "DP1_B", "FGF1_B" } ]),
```

```

fr-dp-geometry ("FR2", [{ "DP2_A", "FGF2_A"}, { "DP2_B", "FGF2_B"},
                           { "DP2_C", "FGF2_C"}]),
fr-dp-geometry ("FR3", [{ "DP3_A", "FGF3_A"}, { "DP3_C", "FGF3_C"}]),
}.

```

The possible combinations between the FGFs are $2*3*2 = 12$ in this case. For each possible combination, there are possible combinations of connectivity of FGFs. For example, the FGF1_A, FGF2_A, and FGF3_A are decomposed cells as shown in the Figure 18. If each FGF is supposed to have three interfaces, there are 16 cases of connectivity as shown each connectivity graph and the number under the graph is the possible combinations of interfaces, which satisfy the connectivity. There exist total 1044 possible combinations of connections between the three FGFs with three interfaces. If we consider assembly sequence in addition to the connections, the number of possible combinations will be increased.

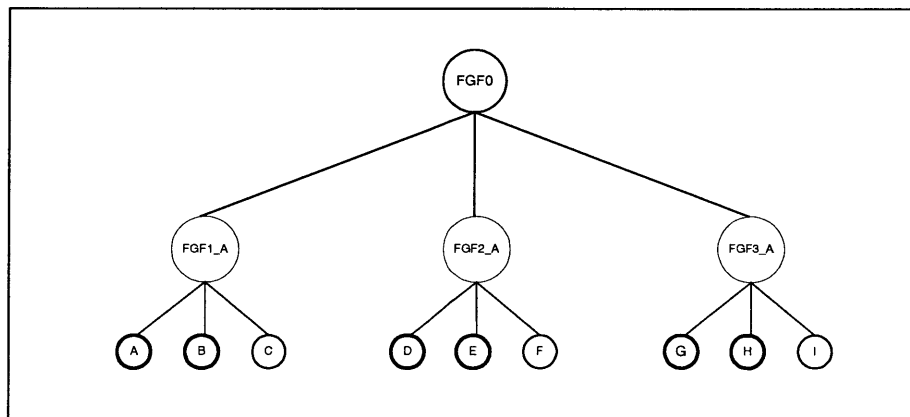
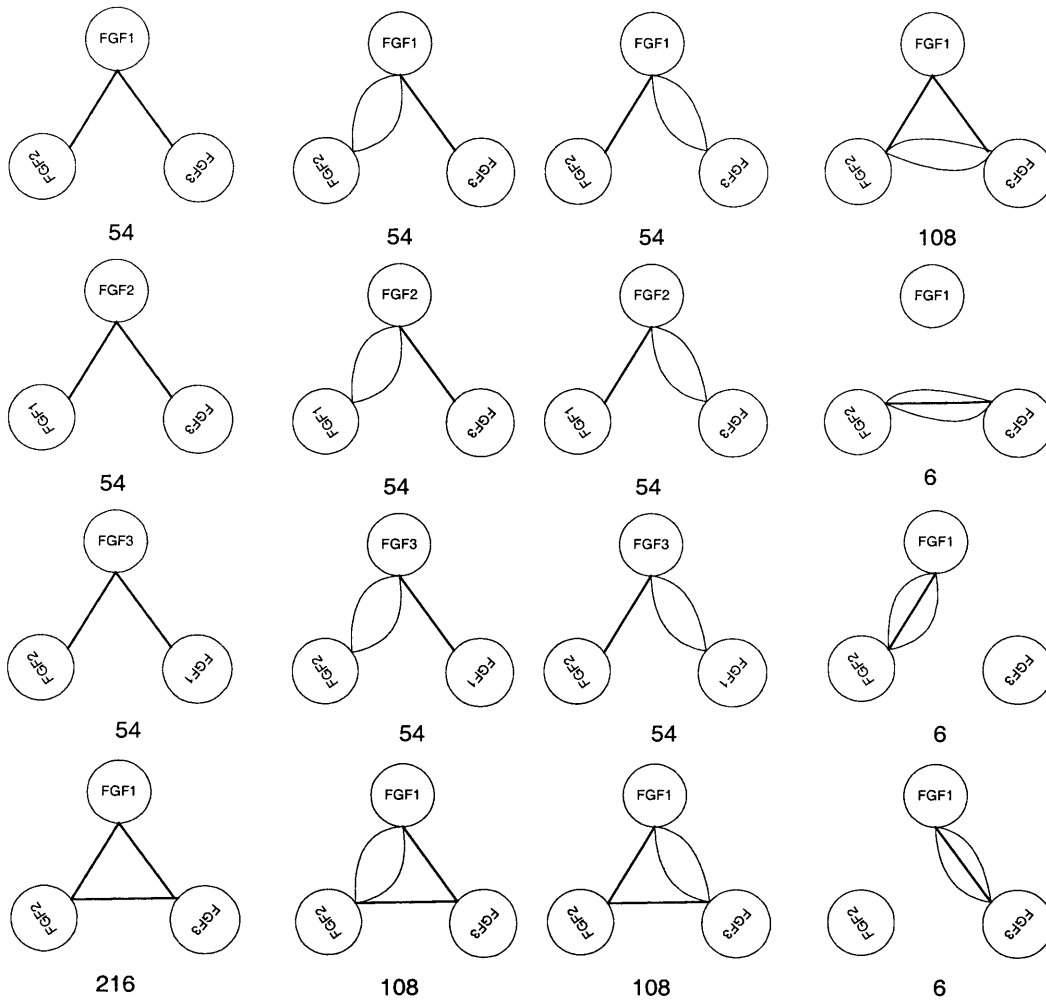


Figure 18. Possible combinations of connectivity of three FGFs in the same level

Because there are a lot of possible combinations for interfacing FGFs, we used GA (Genetic Algorithm) to avoid checking all the combinations described above instead of using an algorithm for generating all the combinations. GA evolution is used to generate assembly sequences between FGFs and to find out combinations of mating faces, which are matched well based on the interface types. GA can evolve to reduce the solution space with an acceptable speed to maximize the fitness. The GA chromosome is defined as

FGFs : 1_A 2_A 2_A 3_A 3_A 1_A

Chromosome:

2	0	2	3	4	6
---	---	---	---	---	---

Interface pairs :

Pair1

Pair2

Pair3

The interface pair represents connections and assembly sequences between two FGFs shown on the top of the chromosome. Each interface pair consists of two genes, which are mapped to the encoded mating faces of an FGF. The first FGF in a pair is a base FGF, which is fixed in the physical space for the assembling operation. The second FGF is an interfacing FGF, which is moved and deformed to be interfaced to the base FGF. In Figure 19, a directional graph shows connections and assembly sequences between FGFs, which the chromosome described above.

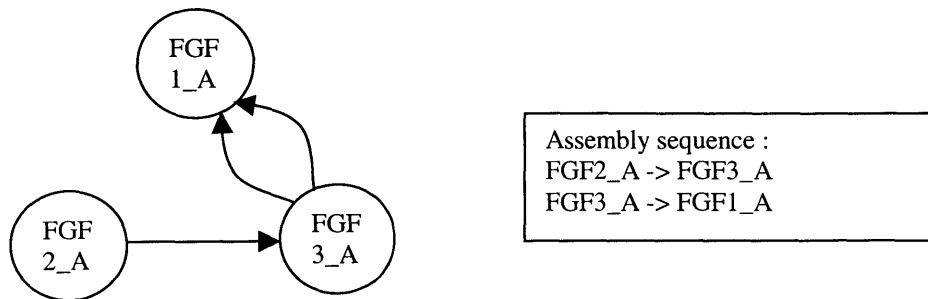


Figure 19. Connections and assembly sequences between FGFs

The number of pairs is determined by ${}_mC_2 = n$, where m is the number of FGFs. Here, the number of pairs in the example chromosome is ${}_3C_2 = 3$. Table 9 shows the encoding of mating faces according to each FGF. Each FGF is supposed to have three mating faces. The codes are generated by combinations of three mating faces in each FGF and are represented as a number in the gene. A number for each FGF is 0, or between 1 and 7. The maximum number of codes for each FGF is determined by ${}_mC_1 + {}_mC_2 + \dots + {}_mC_m$. Thus, the chromosome with codes (2 0 2 3 4 6) represents one possibility of the eleventh connectivity graph and the assembly sequence shown in Figure 19. In this example, each pair can have any two numbers between 0 and 7. If 0 is selected for a gene in a pair, there is no connection between two FGFs in the pair.

FGF	Code	Mating faces	FGF	Code	Mating faces	FGF	Code	Mating faces
	0	Null						
1_A	1	A	2_A	1	D	3_A	1	G
	2	B		2	E		2	H
	3	C		3	F		3	I
	4	A & B		4	D & E		4	G & H
	5	B & C		5	E & F		5	H & I
	6	C & A		6	F & G		6	I & G
	7	A & B & C		7	D & E & F		7	G & H & I

Table 9. Encoding of interfaces of FGFs into genes

Based on the chromosome with codes (2 0 2 3 4 6), the combinations of mating faces and assembly sequences are generated. Table 10 shows the combinations of the mating faces and the assembly sequences of FGFs. The sequence of the genes represents assembly sequences of FGFs. There exists only one possible combination of mating faces between FGF_2 and FGF_3. It is to interface E and I. However, there are two combinations of mating faces between FGF_3 and FGF_1 as shown in Table 10. The assembly sequences of mating

faces in a pair are not considered. It means that either assembly sequence of mating faces, G:C -> H:A or H:A-> G:C is arbitrarily selected for checking geometric interface-ability.

FGF1	FGF_2	FGF_2	FGF_3	FGF_3	FGF_1
2	0	2	3	4	6
No interface		E:I		G:C & H:A	
				H:C & G:A	

Table 10. Combinations of interfacing faces represented by a chromosome (2 0 2 3 4 6)

Through the GA evolutions, we can rank the fitness of interface type (IT) matching between FGFs. The high fitness valued chromosome represents the high possibility of interface type matching of the mating faces between all the FGFs. The value, 1.0, is assigned for the fitness, if interface types of all the mating faces of the FGFs are completely matched each other. The fitness can be represented as

$$fitness = IT_metric = \frac{2 \times \# \text{ of matched ITs}}{\# \text{ of mating faces}} \quad (b)$$

After finishing GA evolution, a shape integration algorithm checks the geometric interface-ability between FGFs from the highest valued chromosome, and generates assembled shapes based on the mating conditions and the generated assembly sequences by GA evolution. The highest fitness valued chromosome has the highest possibility to geometric interface-ability of FGFs, because it gives the best combinations for interface type matching. The fitness values, already decided by GA, are degraded, or maintained by checking geometric interface-ability. The geometric interface-ability between FGFs is quantified as INTERFACE_metric. Thus, if all the mating faces of all the FGFs are completely assembled in the physical space satisfying all the mating conditions, the INTERFACE_metric gives a value of 1.0. If no mating faces can be interfaced, the INTERFACE_metric value will be 0.0. For example of Figure 18, the maximum value of INTERFACE_metric is 0.89 ($=2 \times 4/9$), when four pairs of mating faces

can be interfaced in 9 mating faces of all three FGFs. Figure 20 show the whole process to calculate the INTERFACE_metric values.

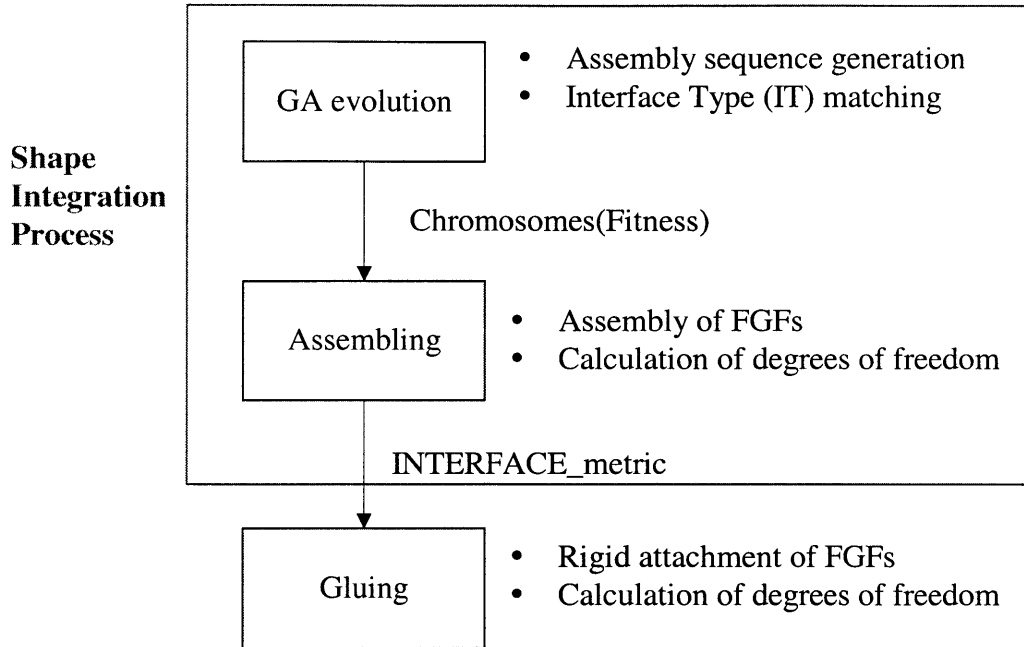


Figure 20. Processes to check interface-ability between FGFs

5.2.2 Automatic Assembly and Interface-ability of FGFs

The goal of the shape integration process is to check geometric interface-ability between FGFs satisfying given mating conditions based on the assembly sequences given from the highest valued chromosomes and finally to quantify the degree of the interface-ability as a value of INTERFACE_metric. This shape integration process has been implemented as computer algorithms that emulates human assembling operations and solves assembly constraint equations using MATLAB. The steps of this process follow the flow chart as shown in Figure 21. The basic strategy is checking the mating conditions by orientating and positioning of FGFs in a given coordinate frame. Here, orienting means only rotations of FGFs to match directions of mating faces, and positioning means only translations

of FGFs to locate the mating faces to the given position. The first FGF in a pair of a chromosome is a base part that is fixed in the absolute coordinate frame. The second FGF is an interfacing part to the base FGF.

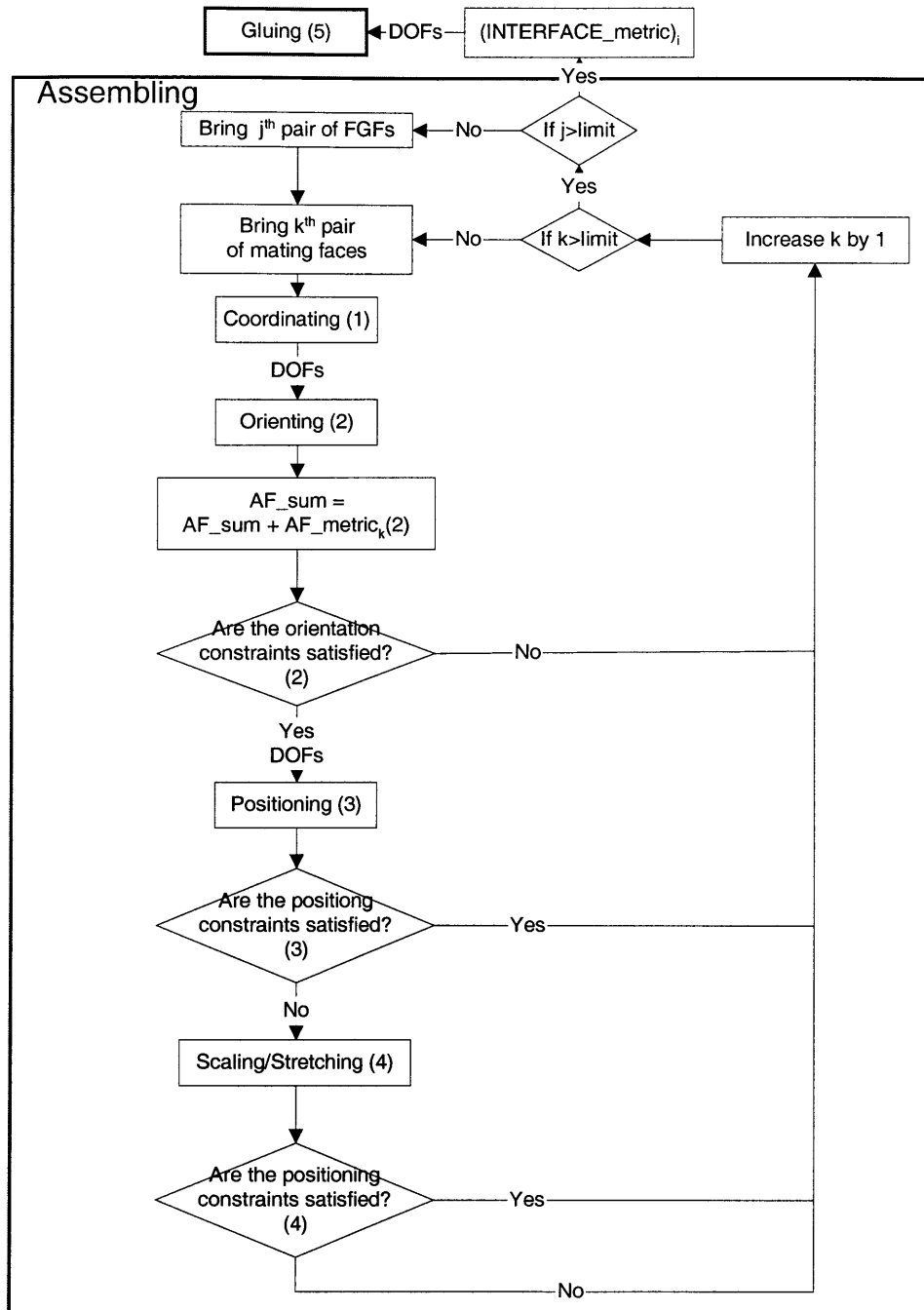


Figure 21. Shape integration process to check geometric interface-ability

The algorithm brings chromosomes sequentially from the highest value to the lower value. In a chromosome, all the combinations of mating faces are generated according to the

codes of the genes. For example, there are two combinations of interfacing mating faces in the example chromosome in section 5.2.1. If the codes of the chromosome are (2 0 4 6 4 6) according to the codes in Table 9, there are four combinations as shown in Table 11. There are two pairs of mating faces in pair 2 and 3 of FGFs.

Sequence	FGF1	FGF_2	FGF_2	FGF_3	FGF_3	FGF_1
	2	0	4	6	4	6
1	No interface		D:I -> E:G		G:C -> H:A	
2	No interface		D:G -> E:I		G:C -> H:A	
3	No interface		D:I -> E:G		H:C -> G:A	
4	No interface		D:G -> E:I		H:C -> G:A	

Table 11. Combinations of interfacing mating faces in a chromosome

The algorithm recursively brings i^{th} sequence ($i = 1, 2, 3, 4$) and j^{th} pair ($j = 1, 2, 3$) in a chromosome, and k^{th} interfacing pair of mating faces ($k = 1, 2$) in pair 2 and pair 3 until all the combinations have been checked. In a pair, the first FGF is fixed as a base part and the second FGF is being interfaced to the first FGF to satisfy pre-defined mating conditions. Each FGF has its local coordinate frame. Thus, the first step of assembling operations is to locate the local coordinate frame with respect to the absolute coordinate frame commonly used for the whole assembling operations. This is called ‘coordinating’ in this thesis. Once the coordinating operation has been performed, the algorithm rotates the second FGF to match orientation of its mating face with that of the base FGF. This operation is called ‘orienting.’ Then, it translates the second FGF to locate its mating face to a position provided by assembly constraints. This operation is called ‘positioning.’ All the operations are performed based on the local coordinate frame and the corresponding DOFs. The mating conditions can be interpreted as assembly constraints described by mathematical equations.

For example of Table 11, the algorithm recursively brings assembly sequences, and automatically assembles the three FGFs. In sequence 1, it brings 2nd pair of FGFs first,

because there is no interface between FGF1 and FGF2. Then, it brings 1st pair of mating faces, D:I. It firstly locates the local coordinate frames of the FGF2 and the FGF3 coincidentally with the absolute coordinate frame. Then, it rotates the FGF3 to a direction to match the orientation of I with that of D according to the given interface type. In the orienting operation, unit normal vectors of two planar mating faces must be parallel and opposite for ‘against’ and two centerlines must be parallel for ‘fits’. The angle differences between mating faces are calculated for AF_metric. It translates the FGF3 for positioning I to D after orienting operations and calculation of values for AF_metric. In the positioning operation, a point on the planar mating face must be on the other planar mating face for ‘against’ and two centerlines must be located on a line for ‘fits’. If scaling or stretching is needed, it is performed. If it fails positioning FGF3, it gives a defined value (0.5 in this thesis) less than 1.0 for Pos_metric. The algorithm output position and orientation of a local coordinate frame w.r.t the absolute coordinate frame, in which the FGF3 can move further with DOFs calculated through the assembling operations. It recursively checks the interface-ability of the second pair of mating faces, E:G, and then do the same steps for 3rd pair of FGFs (FGF3 and FGF1). Finally, three values of IT_metric determined by GA evolutions, AF_metric, and Pos_metric are multiplied as a value of INTERFACE_metric, which represents the geometric interface-ability for a given sequence in a chromosome. The value of the INTERFACE_metric has scaled between 0.0 and 1.0. 1.0 means complete interface-ability for the given FGFs and 0.0 means that any of the given FGFs cannot be interfaced. Consequently, the proposed algorithm produces the assembled shapes and degrees of freedoms of each FGF in the assembled shape, and ranks the assembled shapes by values of INTERFACE_metric, which represent degrees of interface-ability of FGFs in the assembled shapes.

All the operations are performed by transformation matrix [34] and the corresponding mathematical equations represented as follows.

$$T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_R = \begin{bmatrix} n_x & o_x & a_x & 0 \\ n_y & o_y & a_y & 0 \\ n_z & o_z & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_T = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (c)$$

, where T_R denotes only rotation and T_T denotes only translation. The elements of T_R matrix always have the following relations represented as six mathematical equations.

$$\begin{aligned}
 n_x^2 + n_y^2 + n_z^2 &= 1 \\
 o_x^2 + o_y^2 + o_z^2 &= 1 \\
 n_x o_x + n_y o_y + n_z o_z &= 0 \\
 a_x &= n_y o_z - n_z o_y \\
 a_y &= o_x n_z - n_x o_z \\
 a_z &= n_x o_y - n_y o_x
 \end{aligned} \tag{d}$$

DOFs are represented by $T_R(n)$ or $T_T(n)$. Here, n is a number less than or equal to three which represent degrees of freedom for rotation or translation. If an FGF has $[T_R(1), T_T(2)]$, it has one DOF for rotation and two DOFs for translation.

Scaling or stretching is performed for special cases to show its effects to generating assembled shapes. The effects or importance of scaling or stretching will be discussed in a later section. An FGF is originally created with its own purpose. Thus, the more the shape of the FGF deforms, the more its purpose is lost. In this sense, only the severe shape deformation is not allowed, but scaling or stretching is performed. Here, scaling of shapes is enlarging or reducing the size of the shapes in all directions in the physical space and stretching is doing it in one direction. The basic scaling matrix is a special form of the transformation matrix, which is

$$D = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{e}$$

where S_x , S_y , and S_z are scaling factors. If all S_x , S_y , and S_z are equal, this matrix can be used for scaling. Otherwise, one of S_x , S_y , and S_z is not zero, it is used for stretching.

(1) Coordinating

Coordinating is an operation to determine a base coordinate frame in which the interfacing FGF can be rotated and translated. A proper coordinate frame is determined from the previous assembling operation, because the assembling operations are procedural and reduce DOFs of the interfacing FGF in this thesis differently from the approach in [31]. The more mating faces between two FGFs exist, the possibility of less DOFs will be increased. A base coordinate frame must be attached to a proper position with proper orientations to correctly represent rotation and translation of the interfacing FGF for the next assembling operation. There are mainly three types of base coordinate frames used in the shape integration process.

- **Coordinate_absolute:** an absolute coordinate frame, which is attached and fixed to the ground.
- **Coordinate_against:** a base coordinate frame which is attached to the point on the first planar mating face of the base FGF that has been already interfaced by an assembling operation of ‘against’ type. The z-axis is coincident with the unit normal vector of the first mating face of the base FGF, N_{b1} , which has been already used for interface. The y-axis follows an intersection line of two mating planes of the base FGF. The origin is any point on the planar mating face of the base FGF.

$$\begin{aligned}\hat{z}^B &= \hat{N}_{b1}^A \\ \hat{y}^B &= \frac{\hat{z}^B \times \hat{N}_{b2}^A}{|\hat{z}^B \times \hat{N}_{b2}^A|} \\ \hat{x}^B &= \hat{y}^B \times \hat{z}^B\end{aligned}$$

- **Coordinate_fits:** a base coordinate frame which is attached to the point of the centerline, C_{b1} , of the base FGF that has been already interfaced by an assembling

operation of ‘fits’ type. The point is common for both a cylinder and a hole. The z-axis is coincident with the centerline of the first mating face of the base FGF, which has been already used for interface.

$$\hat{z}^B = \frac{\vec{C}_{b1}^A - \vec{C}_{b2}^A}{|\vec{C}_{b1}^A - \vec{C}_{b2}^A|}$$

A transformation matrix from an absolute coordinate frame to a base coordinate frame is obtained by the following method.

```
Coord_abs_base( ) {
   $\hat{x}^A = (1,0,0)$ ,  $\hat{y}^A = (0,1,0)$ ,  $\hat{z}^A = (0,0,1)$ ,  $\vec{O}^A = (0,0,0)$ 
   $\hat{x}^B = [T_{RC}^B] \hat{x}^A$ ,  $\hat{y}^B = [T_{RC}^B] \hat{y}^A$ ,  $\hat{z}^B = [T_{RC}^B] \hat{z}^A$ 
   $\vec{O}^B = [T_{TC}^B] \vec{O}^A$ 
   $[T_C^B] = [T_{TC}^B] [T_{RC}^B]$ 
}
```

(2) Orienting

Orienting operation only rotates the interfacing FGF to match a direction of its mating face with a direction of the other mating face of the base FGF according to the given mating conditions. The followings are the methods necessary for the orienting operations. The superscript denotes a coordinate frame. Here, ‘A’ is the absolute coordinate frame, ‘B’ is a base coordinate frame, Lb is the original local coordinate frame of the base FGF, and Li is the original local coordinate frame of the interfacing FGF.

```
Orienting_against (TR(3)) {
  Inputs :  $\hat{N}_b^{Lb}, \hat{N}_i^{Li}$ 
  Outputs :  $[T_{Ri}^A]$ 
  Equations : equations (d), and
```


$$\begin{aligned}
\hat{N}_b^A &= [T_b^A] \hat{N}_b^{Lb} \\
\hat{N}_i^A &= [T_{Ri}^A] \hat{N}_i^{Li} \\
\hat{N}_i^A &= -\hat{N}_b^A
\end{aligned} \tag{f}$$

where N_b is a unit normal vector of a mating face of a base FGF and N_i is a unit normal vector of a mating face of an interfacing FGF.

}

Orienting_fits ($T_R(3)$) {

Inputs : $\vec{C}_{b1}^{Lb}, \vec{C}_{b2}^{Lb}, \vec{C}_{i1}^{Li}, \vec{C}_{i2}^{Li}$

Outputs : $[T_{Ri}^A]^{(1)}$ or $[T_{Ri}^A]^{(2)}$

Equations : equations (d), and

$$\begin{aligned}
\vec{C}_{b1}^A &= [T_b^A] \vec{C}_{b1}^{Lb} \\
\vec{C}_{b2}^A &= [T_b^A] \vec{C}_{b2}^{Lb} \\
\vec{C}_{i1}^A &= [T_{Ri}^A] \vec{C}_{i1}^{Li} \\
\vec{C}_{i2}^A &= [T_{Ri}^A] \vec{C}_{i2}^{Li}
\end{aligned} \tag{g}$$

$$(C_{i2,x}^A - C_{i1,x}^A) / (C_{b2,x}^A - C_{b1,x}^A) = (C_{i2,y}^A - C_{i1,y}^A) / (C_{b2,y}^A - C_{b1,y}^A) = (C_{i2,z}^A - C_{i1,z}^A) / (C_{b2,z}^A - C_{b1,z}^A) \tag{1}$$

$$(C_{i1,x}^A - C_{i2,x}^A) / (C_{b2,x}^A - C_{b1,x}^A) = (C_{i1,y}^A - C_{i2,y}^A) / (C_{b2,y}^A - C_{b1,y}^A) = (C_{i1,z}^A - C_{i2,z}^A) / (C_{b2,z}^A - C_{b1,z}^A) \tag{2}$$

where C is a point on the centerline of a cylindrical surface. C_{b1} and C_{b2} are two points on the centerline of a base FGF and C_{i1} and C_{i2} are two points on the centerline of an interfacing FGF.

}

Orienting_any ($T_R(1)$) { //rotation about z axis, \hat{z}^B , in a base coordinate frame

Inputs : \hat{N}_b^B, \hat{N}_i^B

Outputs : $[T_{Ri}^B]$

Equations :

$$\begin{bmatrix} \frac{N_{b,x}^B}{\sqrt{(N_{b,x}^B)^2 + (N_{b,y}^B)^2}} \\ \frac{N_{b,y}^B}{\sqrt{(N_{b,x}^B)^2 + (N_{b,y}^B)^2}} \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} \frac{N_{i,x}^B}{\sqrt{(N_{i,x}^B)^2 + (N_{i,y}^B)^2}} \\ \frac{N_{i,y}^B}{\sqrt{(N_{i,x}^B)^2 + (N_{i,y}^B)^2}} \end{bmatrix} \quad (h)$$

$$[T_{Ri}^B] = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where N_b is the unit normal vector of the mating face of the base FGF and N_i is the unit normal vector of the mating face of the interfacing FGF.

}

Originally, a rigid body has three rotational DOFs in 3-dimensional spaces. To satisfy a pre-defined interface type, ‘against’ or ‘fits’ in the orienting operations, `orienting_against` ($T_R(3)$) or `orienting_fits` ($T_R(3)$) are used. `Orienting_against` ($T_R(3)$) receives unit normal vectors of the mating faces as inputs and calculates a rotational transformation matrix. (f) has 6 equations for 12 unknowns. Additional 6 equations are provided by (d). `Orienting_fits` ($T_R(3)$) receives two points on a centerline and two points on the other centerline, and calculates two rotational transformation matrices in two different cases. In the first case, a direction of a vector from C_{b2} to C_{b1} is the same as that of a vector from C_{i2} to C_{i1} . In the second case, both directions are opposite. For both cases, (g) have 9 equations for 15 unknowns. (d) are the additional 6 equations.

Once ‘against’ or ‘fits’ has been applied, the rotational DOFs is reduced to one for all the cases. The rotational axis for the remaining DOF is defined as z-axis of the base coordinate frame, `coordinate_against` or `coordinate_fits`. From the second pair of mating faces, the interfacing FGF rotates about the z-axis. Figure 22 shows how `orienting_any` ($T_R(1)$) calculates a rotational transformation matrix from input unit normal vectors and the rotation of the interfacing FGF. N_b^k is a unit normal vector of k^{th} mating face of base FGF and N_i^k is that of interfacing FGF. Those two vectors are projected on the xy-plane and then

normalized. The corresponding projected and normalized vector, N_{ip}^k , rotated until it is matched with $-N_{bp}^k$. Equations (h) calculate the rotational angle for T_{Ri}^B . This concept can be directly applied for ‘against’ and the projected normal vectors can be considered as normalized vectors from C_{b2} to C_{b1} and from C_{i2} to C_{i1} for ‘fits’.

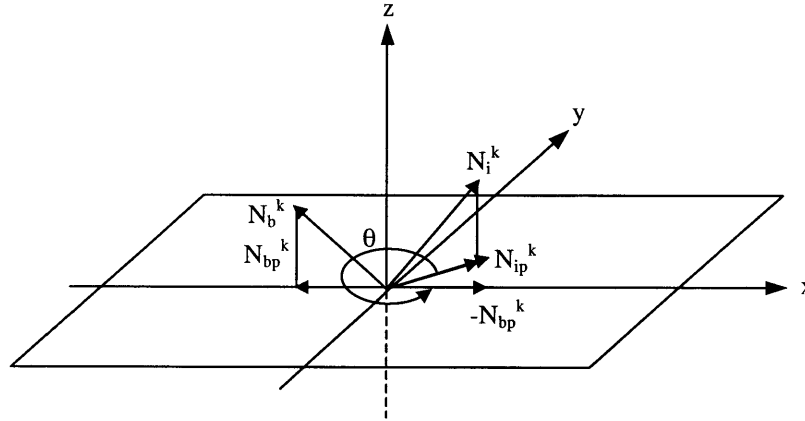


Figure 22. Orienting of the interfacing FGF

AF_metric measures the orient-ability of faces based on the angles between the faces. It quantifies the orient-ability by the following equations.

$$AF_metric_p^k = \sum_{p=1}^{k-1} \left(1 - \frac{\sqrt{\min(\angle_b - \angle_i)^2}}{360} \right)_{pk}, \quad \text{if } k \geq 2 \quad (i)$$

$$AF_metric = \frac{\sum_{j=1}^n \left(\sum_{k=2}^m AF_metric_p^k \right)_j}{\sum_{j=1}^n ({}_m C_2)_j}$$

where k is k^{th} pair of mating faces, j is j^{th} pair of FGFs, m is the total number of pairs of mating faces between two FGFs, and n is the number of FGFs. In the $AF_metric_p^k$ for the k^{th} pair of mating faces, \angle_b is an angle between two normal vectors of two mating faces on

the base FGF, and $angle_i$ is an angle between two normals of the two mating faces on the interfacing FGF. Either angle can be calculated using the mathematical equations, $angle_b = \cos^{-1}(\hat{N}_b^p \bullet \hat{N}_b^k)$ and $angle_i = \cos^{-1}(\hat{N}_i^p \bullet \hat{N}_i^k)$, where N_b is a unit normal of the base FGF and N_i is a unit normal of the interfacing FGF. The arc cosine gives two values. Thus, the minimum value of ($angle_b - angle_i$) among the 4 combinations is selected in AF_metric , because it has been already rotated to the orientation in which the angle between both mating faces has been minimized. If it is 0° , orientations of both FGFs are completely matched. If k is greater than or equal to three, $k-1$ combinations, in which the given pair of mating faces is combined to already interfaced pairs, are checked to calculate and sum the scaled values from 0.0 to 1.0. The values of $AF_metric_p^k$ for k^{th} pair are averaged to AF_metric for the given pair of interfacing FGFs as shown in the flow chart in Figure 21. For example, Figure 23 shows the simple calculation of $AF_metric_1^2$ for two pairs of mating faces.

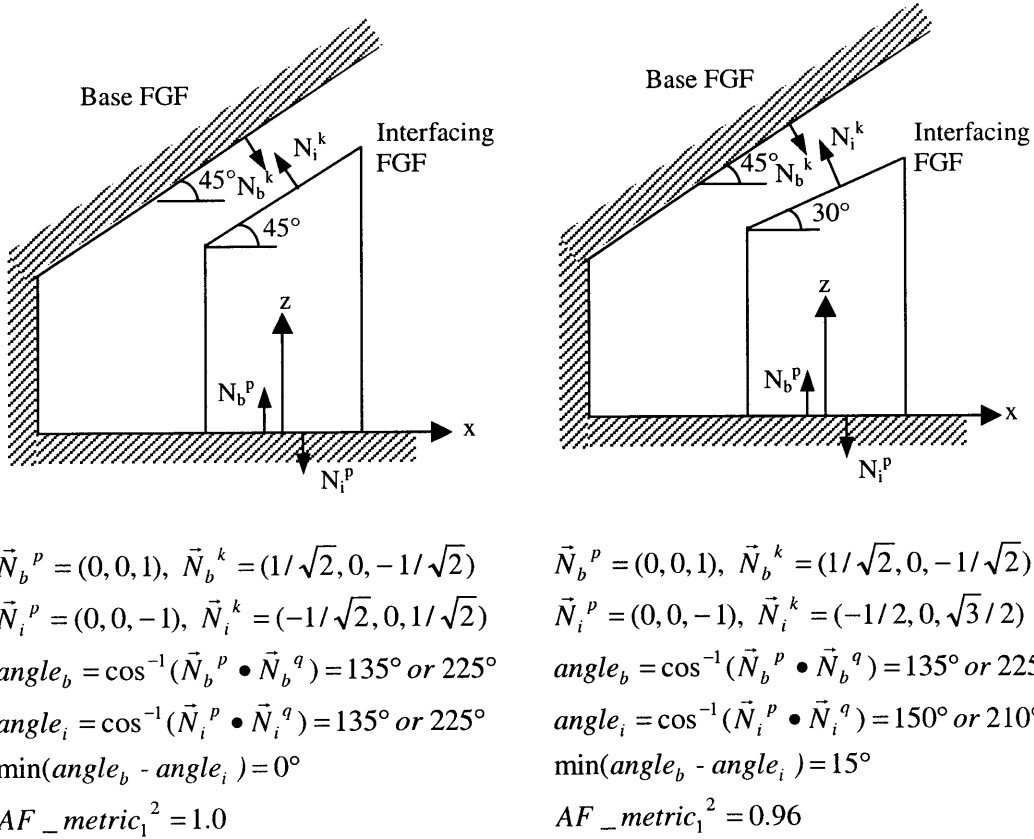


Figure 23. Calculation of AF_metric

(3) Positioning

Positioning operation only translates the interfacing FGF to locate it on the correct position by which the given mating conditions are satisfied. The followings are the methods for the positioning operations based on the DOFs given from the preceded assembling operations.

Positioning_against ($T_T(3)$) {

Inputs : $\hat{N}_b^{Lb}, \vec{P}_b^{Lb}, \vec{P}_i^{Li}, [T_{Ri}^A]$

Outputs : $[T_{Ti}^A]$

Equations :

$$\begin{aligned}\hat{N}_b^A &= [T_b^A] \hat{N}_b^{Lb} \\ \vec{P}_b^A &= [T_b^A] \vec{P}_b^{Lb} \\ \vec{P}_i^A &= [T_{Ti}^A][T_{Ri}^A] \vec{P}_i^{Li} \\ \vec{P}_i^A &= \vec{P}_b^A\end{aligned}\quad (j)$$

where P_b is a point on a mating face of a base FGF and P_i is a point on a mating face of an interfacing FGF.

}

Positioning_fits ($T_T(3)$) {

Inputs : $\vec{C}_{b1}^{Lb}, \vec{C}_{i1}^{Li}, \vec{C}_{i2}^{Li}, [T_{Ri}^A]^{(1)}, [T_{Ri}^A]^{(2)}$

Outputs : $[T_{Ti}^A]^{(1)}, [T_{Ti}^A]^{(2)}$

Equations :

$$\begin{aligned}\vec{C}_{b1}^A &= [T_b^A] \vec{C}_{b1}^{Lb} \\ \vec{C}_{i1}^A &= [T_{Ti}^A]^{(1)} [T_{Ri}^A]^{(1)} \vec{C}_{i1}^{Li} \text{ or } \vec{C}_{i1}^A = [T_{Ti}^A]^{(2)} [T_{Ri}^A]^{(2)} \vec{C}_{i1}^{Li} \\ \vec{C}_{i1}^A &= \vec{C}_{b1}^A\end{aligned}\quad (k)$$

where C is a point on the centerline of a cylindrical surface. C_{b1} is a point on the centerline of a base FGF and C_{i1} and C_{i2} are two points on the centerline of an interfacing FGF. (1) or (2) is either case of orienting.

}

Positioning_against (T_T(2)) {

Inputs : $\hat{N}_b^A, \vec{P}_b^A, \vec{P}_i^A, [T_C^B], [T_{Ri}^B]$

Outputs : $[T_{Ti}^B]$

Equations :

$$\hat{N}_b^B = [T_{Ri}^B][T_C^B]\hat{N}_b^A$$

$$\vec{P}_b^B = [T_{Ri}^B][T_C^B]\vec{P}_b^A$$

$$Plane : N_{b,x}^B x + N_{b,y}^B y + N_{b,z}^B z + d = 0$$

$$d = -N_{b,x}^B P_{b,x}^B - N_{b,y}^B P_{b,y}^B - N_{b,z}^B P_{b,z}^B$$

$$\vec{P}_i^B = [T_{Ti}^B]^* [T_{Ri}^B][T_C^B]\vec{P}_i^A$$

$$[T_{Ti}^B]^* = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

$$N_{b,x}^B P_{i,x}^B + N_{b,y}^B P_{i,y}^B + N_{b,z}^B P_{i,z}^B + d = 0$$

$$[T_{Ti}^B] = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where P_b is a point on a mating face of a base FGF and P_i is a point on a mating face of an interfacing FGF.

}

Positioning_fits (T_T(2)) {

Inputs : $\vec{C}_{b1}^A, \vec{C}_{b2}^A, \vec{C}_{i1}^A, \vec{C}_{i2}^A, [T_C^B], [T_{Ri}^B]$

Outputs : $[T_{Ti}^B]$

Equations :

$$\vec{C}_{b1}^B = [T_{Ri}^B][T_C^B]\vec{C}_{b1}^A, \vec{C}_{b2}^B = [T_{Ri}^B][T_C^B]\vec{C}_{b2}^A$$

$$\vec{C}_{i1}^B = [T_{Ri}^B][T_C^B]\vec{C}_{i1}^A, \vec{C}_{i2}^B = [T_{Ri}^B][T_C^B]\vec{C}_{i2}^A$$

$$\hat{z}^B \bullet \vec{P}_b^B = -p$$

$$\frac{\vec{C}_{b1}^B - \vec{C}_{b2}^B}{|\vec{C}_{b1}^B - \vec{C}_{b2}^B|} \bullet \vec{P}_b^B = 0$$

$$\hat{z}^B \bullet \vec{P}_i^B = -p$$

$$\frac{\vec{C}_{i1}^B - \vec{C}_{i2}^B}{|\vec{C}_{i1}^B - \vec{C}_{i2}^B|} \bullet \vec{P}_i^L = 0$$

$$\vec{P}_b^L = [T_{Ti}^L]\vec{P}_i^L, \quad [T_{Ti}^L] = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (m)$$

where P_b is an intersection point vector between xy-plane and a centerline of the base FGF and P_i is an intersection point vector between xy-plane and a centerline of the interfacing FGF.

}

Positioning_any ($T_T(1)$) {

Inputs : an axis, $\vec{P}_b^A, \vec{P}_i^A, [T_C^B], [T_{Ri}^B]$

Outputs : $[T_{Ti}^B]$

Equations :

$$\vec{P}_b^B = [T_{Ri}^B][T_C^B]\vec{P}_b^A$$

$$\vec{P}_i^B = [T_{Ti}^B][T_{Ri}^B][T_C^B]\vec{P}_i^A \quad (n)$$

$$P_{b,any}^B = P_{i,any}^B + p_{any}$$

where P_{any} is the vector component in a given axis of the base coordinate frame.

}

Originally, a rigid body has three translational DOFs in 3-dimensional space. To positioning FGFs with 3 DOFs, positioning_against ($T_T(3)$) or positioning_fits ($T_T(3)$) are used. Positioning_against ($T_T(3)$) receives input normal and input points on the planar mating faces of the FGFs and calculates a translational transformation matrix. Six equations in (j) are solved for 6 unknowns in this method. Positioning_fits ($T_T(3)$) coincides a point on the centerline of the interfacing FGF with that of the base FGF. Six unknowns can be solved using six equations in (k) for either case.

Positioning_against ($T_T(2)$) locates a point on the mating face of the interfacing FGF onto the plane on which the mating face of the base FGF lies. As shown in Figure 23, the interfacing FGF has two translational DOFs in x and y direction, once one ‘against’ assembling operation has been already performed. Because the second pair of the mating faces has been oriented, the interfacing FGF must move in only x direction until its second mating face is located on the plane with N_b^k in the base coordinate frame. Thus, two unknowns, $P_{i,x}^B$ and p_x , can be calculated by the corresponding two equation in (l).

A metric for positioning mating faces is defined as below. Once positioning of a mating face is failed, a value less than 1.0 is assigned to degrade the INTERFACE_metric assigned to quantify interface-ability between FGFs in a chromosome. If it is successful, 1.0 is assigned to position metric for the mating.

$$Pos_metric^k = \begin{cases} 1.0, & \text{if positioning is successful} \\ 0.5, & \text{if positioning is unsuccessful} \end{cases} \quad (i)$$

$$Pos_metric = \frac{\sum_{j=1}^{nC_2} ((\sum_{k=1}^m Pos_metric^k) / m)_j}{nC_2}$$

where k is k^{th} pair of mating faces, j is j^{th} pair of FGFs, m is the total number of mating pairs between two FGFs, and n is the number of FGFs.

(4) Scaling / Stretching

In this thesis, only scaling operation has been applied for some special cases. The first case is for a type of interface, 'tight-fits:assembly'. Once all the assembling operations by 'against' and 'fits' have been done, the algorithm checks the details of the 'fits'. If a 'fits' was 'tight-fits,' it scales the corresponding interfacing FGF in a base coordinate frame with a scaling factor, $S = \text{radius of base cylindrical face} / \text{radius of interfacing cylindrical face}$. The second case is 'against_equal:rigid_attachment'. Though a shape deformation or optimization technique to match boundaries of the mating faces is not considered, a simple scaling for matching circular outer boundaries of two planar mating faces has been tested. It has been performed by additional concentric constraints and the scaling factor, S , the same for 'tight-fits'. The results of this operation will be shown in the section for application examples.

(5) Automatic assembly methods

The methods proposed in coordinating, orienting, positioning operations are the bases to automatically assemble an interfacing FGF to a base FGF. Each method mainly outputs transformation matrices, a base coordinate frame and information of DOFs from the given inputs. The outputs of a method can provide information for inputs to another method. Because assembling operations must be performed following given types of interface in a chromosome, there should not be any discontinuity from outputs of a method to inputs of the other method for automating the assembling operations. All the cases of the information flow for 'against' and 'fits' have been considered, and then 10 methods have been made to be recursively used according to the input type of interface and DOFs. Table 12 summarizes all the methods.

```

Against:assembly or Fits:assembly{
    Coordinating
    Orienting
    Positioning
}

```

Assembly methods(DOFs)	Output DOFs	Conditions
Against:assembly($T_T(3)$, $T_R(3)$) coordinating, orienting, positioning	$T_T(2)$, $T_R(1)$	Always successful
Against:assembly($T_T(2)$, $T_R(1)$) coordinating, orienting, positioning	$T_T(2)$, $T_R(1)$	If the normal has the same direction as the axis of rotation
	$T_T(1)$, $T_R(0)$	If the normal has a different direction from the axis of rotation
	Failed	Scaling / Stretching
Against:assembly($T_T(1)$, $T_R(0)$) coordinating, positioning	$T_T(1)$, $T_R(0)$	If the mating face is parallel to the axis of translation
	$T_T(0)$, $T_R(0)$	If the mating face is not parallel to the axis of translation
	Failed	Scaling / Stretching
Against:assembly($T_T(1)$, $T_R(1)$) coordinating, orienting, positioning	$T_T(1)$, $T_R(1)$	If the normal has the same direction as the axis of rotation, and if the mating face is parallel to the axis of translation
	$T_T(0)$, $T_R(1)$	If the normal has the same direction as the axis of rotation, and if the mating face is not parallel to the axis of translation

	$T_T(1), T_R(0)$	If the normal has a different direction from the axis of rotation, and if the mating face is parallel to the axis of translation
	$T_T(0), T_R(0)$	If this is not the cases above
	Failed	Scaling / Stretching
Against:assembly($T_T(0), T_R(1)$) coordinating, orienting	$T_T(0), T_R(1)$	If the normal has the same direction as the axis of rotation
	$T_T(0), T_R(0)$	If the normal is different from the axis of rotation
	Failed	Scaling / Stretching
Fits:assembly($T_T(3), T_R(3)$) coordinating, orienting, positioning	$T_T(1), T_R(1)$	Always successful
Fits:assembly($T_T(1), T_R(1)$) coordinating, orienting, positioning	$T_T(1), T_R(1)$	If the centerline is the same as the axis of rotation, and parallel to the axis of translation
	$T_T(1), T_R(0)$	If the centerline is different from the axis of rotation, but parallel to the axis of translation
	$T_T(0), T_R(1)$	If the centerline is the same as the axis of rotation, but not parallel to the axis of translation
	$T_T(0), T_R(0)$	If this is not the cases above
	Failed	Scaling / Stretching
Fits:assembly($T_T(1), T_R(0)$) coordinating, positioning	$T_T(1), T_R(0)$	If the centerline is parallel to the axis of translation
	$T_T(0), T_R(0)$	If the centerline is not parallel to the axis of translation
	Failed	Scaling / Stretching

Fits:assembly($T_T(2)$, $T_R(1)$) coordinating, orienting, positioning	$T_T(0)$, $T_R(1)$	If the centerline is the same as the axis of rotation
	$T_T(1)$, $T_R(0)$	If the centerline is parallel to the axis of translation
	$T_T(0)$, $T_R(0)$	If this is not the cases above
	Failed	Scaling / Stretching
Fits:assembly($T_T(0)$, $T_R(1)$) coordinating, orienting	$T_T(0)$, $T_R(1)$	If the centerline is the same as the axis of rotation
	$T_T(0)$, $T_R(0)$	If the centerline is different from the axis of rotation
	Failed	Scaling / Stretching

Table 12. Assembly methods and their output DOFs

An INTERFACE_metric for assembling FGFs based on a chromosome is calculated by

$$INTERFACE_metric = IT_metric \times AF_metric \times Pos_metric .$$

(6) Gluing

This thesis focuses on automatic assembling operations and measures INTERFACE_metric as a degree of satisfaction of pre-defined mating conditions during the automatic assembly operations. Gluing may be performed through a detailed shape deformation process or by a shape optimization technique, once he/she decides the candidate assembled shapes.

5.3 Top-down Decomposition for Combining FGFs

Figure 18 shows the possible combinations of interfacing all three FGFs from the leaf level. Once FGF1_A and FGF2_A are interfaced first on the different branch from that of FGF3_A in a top-down hierarchical tree, the possible combinations of interfaces of three FGFs are reduced to 870. All the cases, in which FGF1_A and FGF2_A were disconnected, have been eliminated from the 1044 combinations. Finally, the 174 combinations can be reduced comparing to 1044. Figure 24 shows the combinations of this case. Thus, the power of V-model design process can be reduction of solution space for combinatorial search. If we rely on only bottom-up approach, the combinatorial complexity will be much higher than that of combinations of top-down decomposition first, search of unknown FGFs, and integration of them as proposed in this thesis. The V-model design process provides designers with the computing environment, in which advantages of both top-down and bottom-up approaches can be maximized.

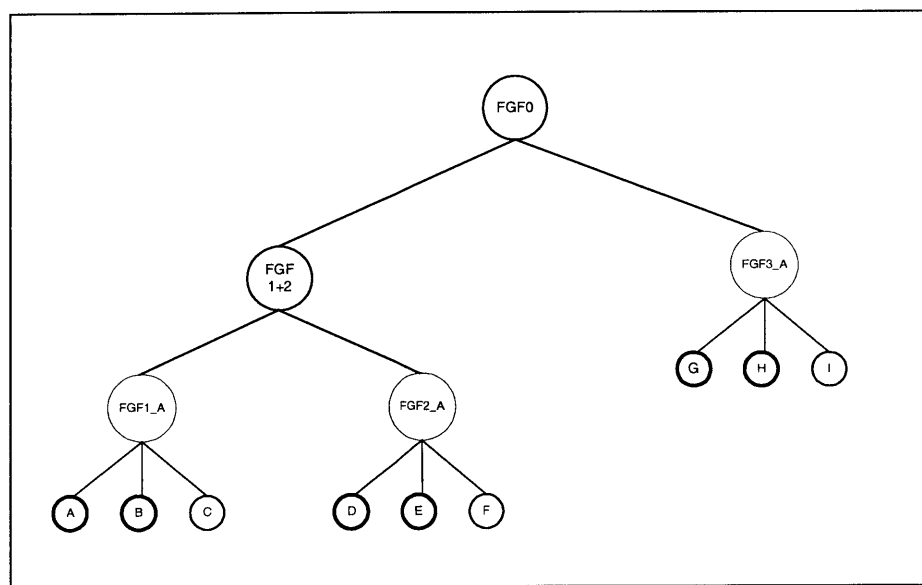
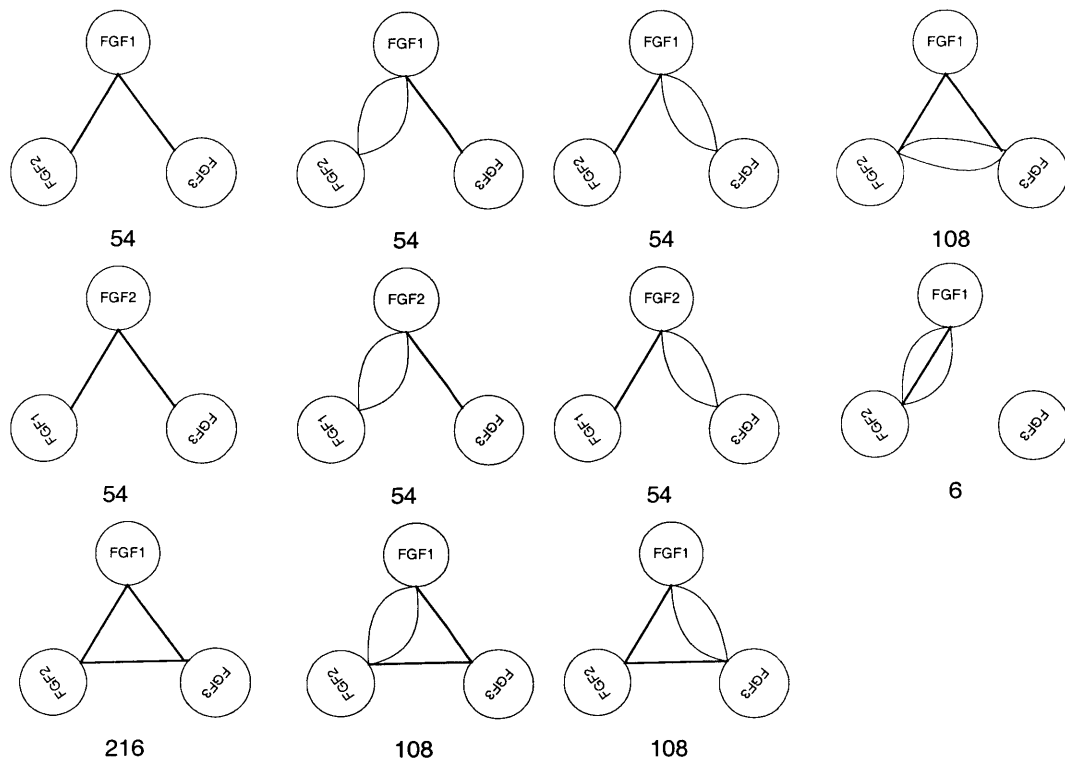


Figure 24. Possible combinations of connectivity of three FGFs in a hierarchical tree

5.4 Shape Determination

The method proposed in this thesis generates many candidate shapes from input functional requirements defined through the V-model design process. Each candidate shape can be selected or discarded by the following three decision supportive data.

- A value of FR_metric: 0.0 ~ 1.0
- A value of INTERFACE_metric: 0.0 ~ 1.0
- Design matrix

The values of FR_metric and INTERFACE_metric are automatically calculated by computer algorithms with the assembled candidate shape. Construction of design matrix for each candidate shape must performed by the designer to check independence axiom. This process is related to change of dimensions of each FGF to satisfy the corresponding FR. The change of dimensions may result in decoupled design matrix to guarantee satisfaction of all the FRs as explained in the hanger design example in the chapter 3.

Chapter 6

Application Examples

All the examples have been made by language matching algorithm and GA implemented using Java and by assembled shape generation algorithm implemented using MATLAB. Database has been constructed by using text files as explained in chapter 4. Assembled shapes have been manually regenerated and visualized using SolidWorks based on the results produced by the algorithms. Figure 25 shows a screen shot of the Java implementations. The minimum value of FR_metric can be set in the textfield labeled Match %. It controls the search space in the database.

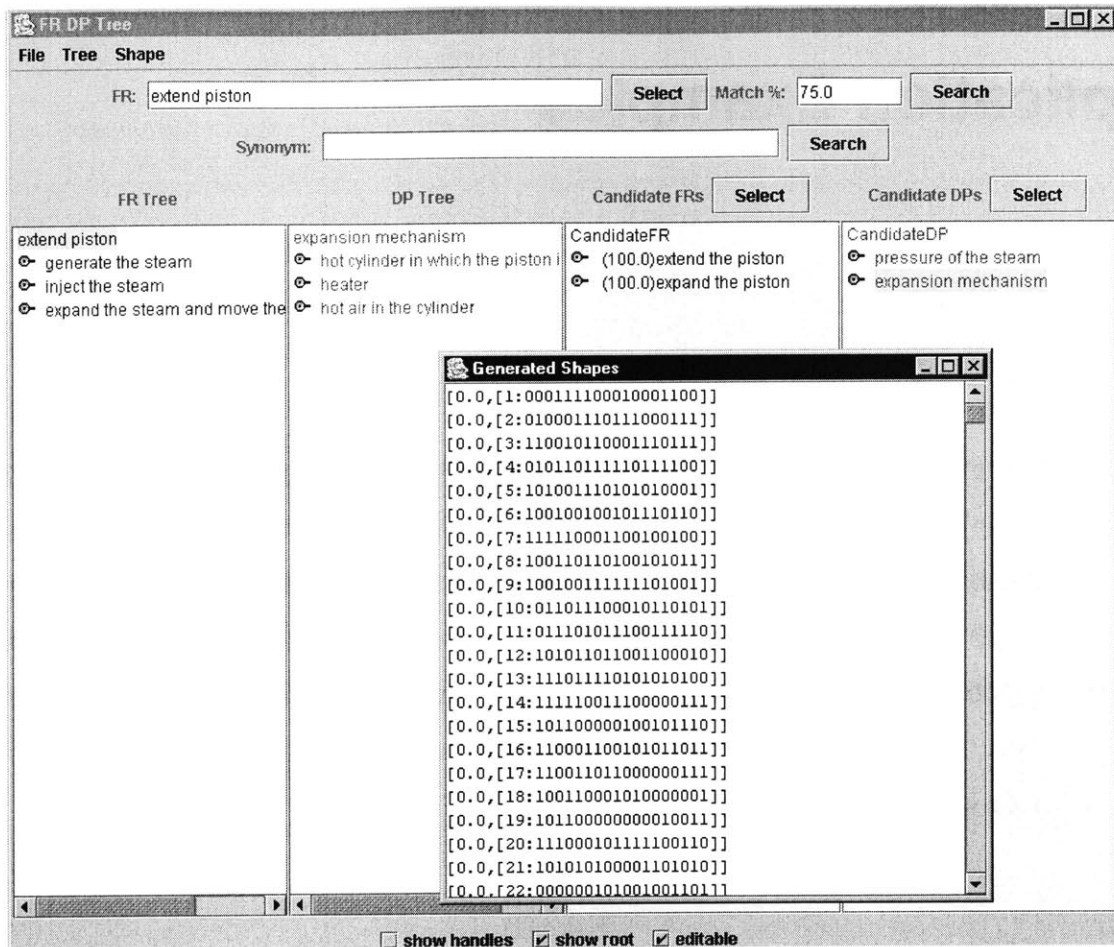


Figure 25. Implementation of FR, DP trees, language matching and GA

6.1 Assembled Shape Generation of two FGFs

This is a simple example to find plausible FGFs from the database and to generate assembled shape based on information of interfaces already stored in the database from the existing design cases. A designer starts a design problem by describing his top-level FR first, think a DP for the top-level FR, and decomposes it into two lower level FRs in the V-model design process. Figure 26 shows the decompositions.

Design Problem

FR0 = contain coffee keeping its temperature warm

FR1 = contain coffee

FR2 = cover DP1

DP0 = container

DP1 = ?

DP2 = ?

Design Case 1

FR0 = contain water and flowers

DP0 = vase

Design Case 2

FR0 = make the people walk on the ground

FR1 = support walking people

FR2 = make people to enter under the road

FR3 = cover the manhole

DP0 = sidewalk road

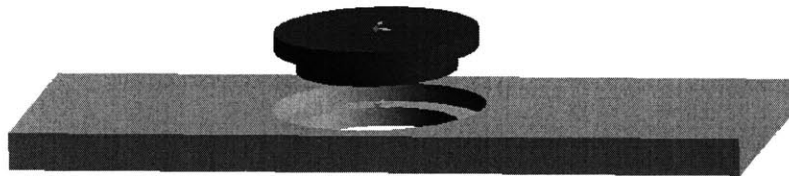
DP1 = road

DP2 = manhole

DP3 = cover of manhole



(a) vase



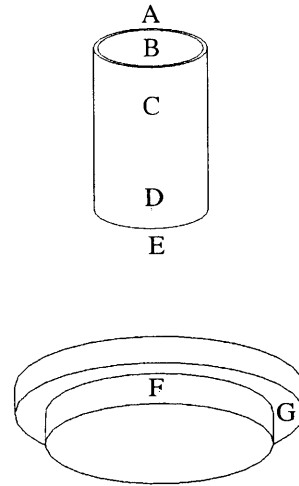
(b) sidewalk road

Figure 26. A simple example for search of proper FGFs and combination of the FGFs

Figure 27 shows the GA encoding as representation of assembly of two pairs of two mating faces between two FGFs. There are 20 combinations of mating faces. The base FGF is the vase and the interfacing FGF is the cover of manhole. The base FGF did not have any pre-defined interface and the interfacing FGF had two pre-defined interfaces. Thus, any two of

five faces on the base FGF are set to be candidate mating faces. This generates 10 combinations (${}_5C_2$) for candidate mating faces. The GA evolution is very quick in 3 generations with 200 populations using simple GA. Only three chromosomes in the population have 1.0 of IT_metric value, because both interface types (against:assembly and tight-fits:assembly) are completely matched in the chromosomes. ‘a’ denotes assembly, ‘c’ denotes cylinder, and ‘h’ denotes hole. In the case of (1 1), there are two combinations for matching interface types, A:F & B:G and B:F & A:G. The combination, B:F & A:G, gives an exact match of the interface types.

FGF	Gene	Mating faces	Interface type
	0	None	
1	1	A & B	Against:a, Tight-fits:ah
	2	A & C	Against:a, Tight-fits:ac
	3	A & D	Against:a, Against:a
	4	A & E	Against:a, Against:a
	5	B & C	Tight-fits:a, Tight-fits:ac
	6	B & D	Tight-fits:ah, Against:a
	7	B & E	Tight-fits:ah, Against:a
	8	C & D	Tight-fits:ac, Against:a
	9	C & E	Tight-fits:ac, Against:a
	10	D & E	Against:a, Against:a
2	1	F & G	Tight-fits:ac, Against:a



1	1
---	---

6	1
---	---

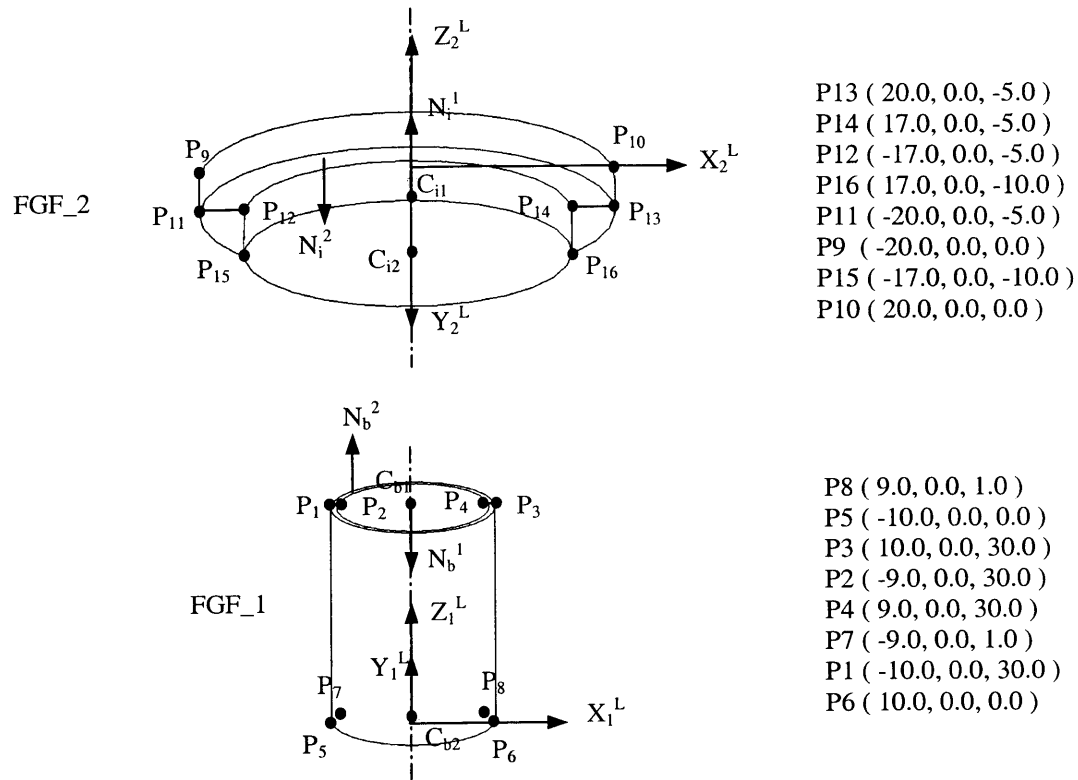
7	1
---	---

FGF1	FGF2	IT metric
1	1	Max:1.0
A:F & B:G		0.0
B:F & A:G		1.0

$$IT_metric = \frac{2 \times \# \text{ of matched ITs}}{\# \text{ of mating faces}} = 1.0$$

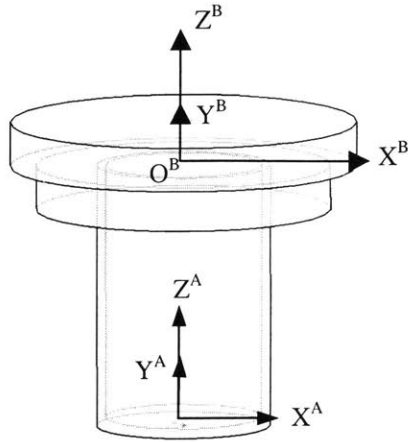
Figure 27. Encoding in GA and calculation of IT_metric

In the next step, the shape integration process automatically assembles two FGFs and checks geometric interface-ability for the three chromosomes based on the AF_metric and Pos_metric. This shape generation process is performed by two assembly methods, fits:assembly ($T_T(3)$, $T_R(3)$) and against:assembly ($T_T(1)$, $T_R(1)$). Figure 28 shows the encapsulated geometric data obtained from STEP files. The table in the Figure also shows data defined for mating faces in chromosome (1 1). Figure 29 shows the results of assembly and calculations of the metrics. It includes value of FR_metric, value of INTERFACE_metric, a base coordinate frame and final DOFs. This information can be used to visualize the assembled shape and analyze the relative motion of the FGFs based on the final DOFs.

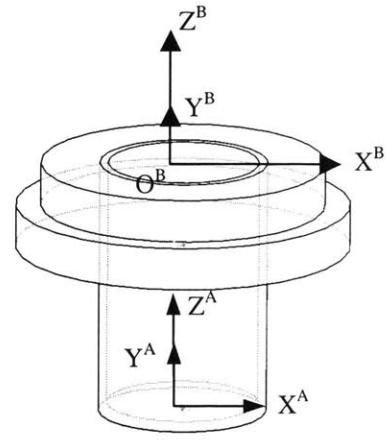


FGF1		FGF2	
Tight-fits:ah	$C_{b1} (0, 0, 30)$	Tight-fits:ac	$C_{i1}(0, 0, -5)$
	$C_{b2} (0, 0, 1)$		$C_{i2}(0, 0, -10)$
Against:a	$P_2 (-9, 0, 30)$	Against:a	$P_{12} (-17, 0, -5)$
	$N_b (1, 0, 0)$		$N_i (1, 0, 0)$

Figure 28. Initial configuration of FGFs and their geometric data



(a) Case I



(b) Case II

FR	DP	FR_metric	INTERFACE_metric
contain coffee	vase	0.4	<u>Case I : $1*1*1=1.0$</u>
cover DP1	cover of manhole	0.5	<u>Case II : $1*0.5*(1+0.5)/2=0.375$</u>

Figure 29. Results of shape generation from two FRs

As shown in the table of Figure 29, fits:assembly deals with two cases as explained in section 4.2.2 (b). Case I can be selected, because its value of INTERFACE_metric is larger than that of case II. In case II, one pair of mating faces cannot be positioned to satisfy the given mating condition, ‘against:assembly,’ based on the DOFs generated by ‘fits:assembly.’ Fits_assembly ($T_T(3)$, $T_R(3)$) method outputs $T_T(1)$, $T_R(1)$ and a base coordinate frame, O^B . The outputs generated by fits_assembly ($T_T(3)$, $T_R(3)$) are provided to against_assembly ($T_T(1)$, $T_R(1)$) method as inputs. Against_assembly ($T_T(1)$, $T_R(1)$) method outputs $T_T(0)$, $T_R(1)$ about the axis, Z^B , of the base coordinate frame, O^B . It means that FGF2 can only rotate about Z^B . The followings are the fits_assembly ($T_T(3)$, $T_R(3)$) and against:assembly ($T_T(1)$, $T_R(1)$) that have been used to get the results.

Fits_assembly ($T_T(3), T_R(3)$) (DOF, $\vec{C}_{b1}^{Lb}, \vec{C}_{b2}^{Lb}, \vec{C}_{i1}^{Li}, \vec{C}_{i2}^{Li}$) {

$[T_{Ri}^A]^{(1)}$ or $[T_{Ri}^A]^{(2)} = \text{Orienting_fits}(\vec{C}_{b1}^{Lb}, \vec{C}_{b2}^{Lb}, \vec{C}_{i1}^{Li}, \vec{C}_{i2}^{Li})$

$[T_{Ti}^A]^{(1)}$ or $[T_{Ti}^A]^{(2)} = \text{Positioning_fits}(\vec{C}_{b1}^{Lb}, \vec{C}_{i1}^{Li}, \vec{C}_{i2}^{Li}, [T_{Ri}^A]^{(1)}, [T_{Ri}^A]^{(2)})$

$\vec{O}^B = \vec{C}_{b1}^A, \hat{z}^B = \frac{\vec{C}_{b1}^A - \vec{C}_{b2}^A}{|\vec{C}_{b1}^A - \vec{C}_{b2}^A|}, \hat{x}^B = \frac{\vec{P}_2^A \times \vec{C}_{b1}^A}{|\vec{P}_2^A \times \vec{C}_{b1}^A|}, \hat{y}^B = \hat{z}^B \times \hat{x}^B$

Return DOF($(\hat{z}^B), (\hat{z}^B)$), $[T_{Ri}^A]^{(1)(2)}, [T_{Ti}^A]^{(1)(2)}, \vec{O}^B, \hat{x}^B, \hat{y}^B, \hat{z}^B$

}

Against:assembly ($T_T(1), T_R(1)$)(DOF($(\hat{z}^B), (\hat{z}^B)$), $[T_{Ri}^A]^{(1)(2)}, [T_{Ti}^A]^{(1)(2)}, \vec{O}^B, \hat{x}^B, \hat{y}^B, \hat{z}^B$) {

$\hat{N}_b^B = [T_C^B] \hat{N}_b^A$

$\hat{N}_i^B = [T_C^B] \hat{N}_i^A$

If($\hat{N}_i^B = \hat{N}_b^B$) AF_metric_p^k = 0.5

Else if($\hat{N}_i^B = -\hat{N}_b^B$) AF_metric_p^k = 1.0

Else {

$[T_{Ri}^B] = \text{Orienting_any}(\hat{N}_b^B, \hat{N}_i^B)$

AF_metric_p^k = AF_metric() //see equations (i)

}

$[T_{Ti}^B] = \text{Positioning_any}(\hat{z}^B, \vec{P}_b^A, \vec{P}_i^A, [T_C^B], [T_{Ri}^B]^{(1)(2)})$

If($\hat{N}_i^B = \hat{z}^B$) $DOF_R = \hat{z}^B$

Else $DOF_R = 0$

If($\hat{N}_i^B \bullet \hat{z}^B = 0$) $DOF_T = \hat{z}^B$

Else $DOF_T = 0$

Return DOF(DOF_T, DOF_R), $[T_{Ri}^B], [T_{Ti}^B]$

}

Figure 30 shows the other results for chromosomes (6 1) and (7 1). Thus, there are two more candidate shapes each of which has 1.0 for INTERFACE_metric. The algorithm proposed in this thesis produces the same results that SolidWorks can produce by manual

assembly modeling operations with the same mating conditions. This algorithm automates the manual assembling operations and quantifies the interface-ability as a value of INTERFACE_metric during the automated assembling operations.

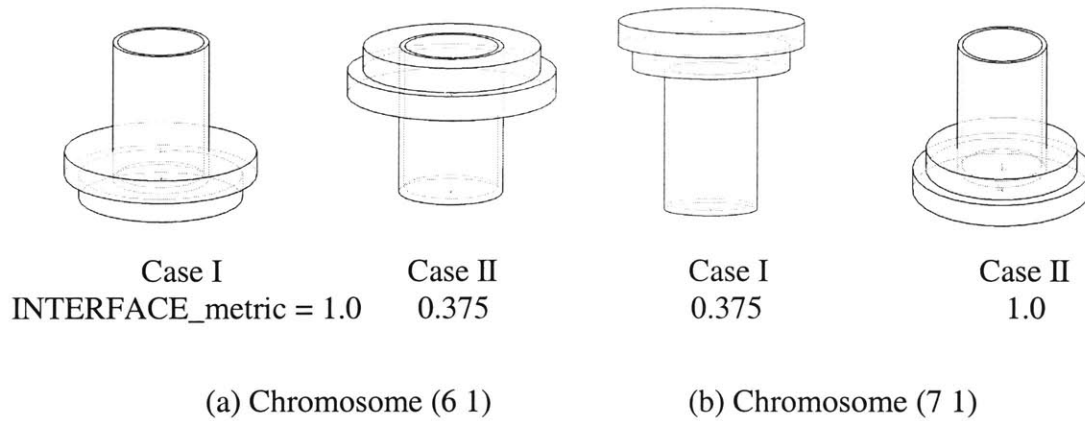


Figure 30. Assembled shapes that chromosomes present

Figure 31 is three resultant shapes, which have the 1.0 value of INTERFACE_metric scaled to satisfy 'tight-fits' instead of only 'fits'. The scaling operation has been performed by calculating radii of two cylindrical faces of cylinder and hole and by making a scaling factor from a ratio of two radii. The FGF2 has one rotational DOF in all the resultant assembled shapes. Actually, FGF2 in (b) and (c) of Figure 31 does not satisfy its functionality, because it does not cover the FGF1. Thus, (a) of Figure 31 must be determined by the human designer as a design solution for the given design problem. In addition, FGF2 interferes with FGF1. If an interference checking routine is implemented, (b) and (c) can be automatically discarded.

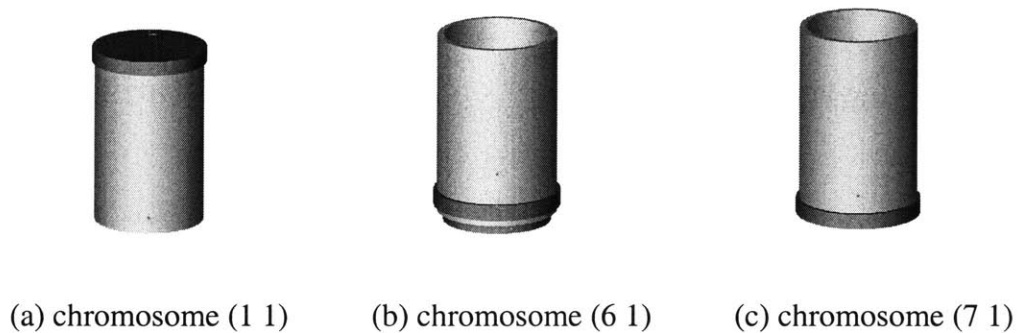


Figure 31. Final results of container design problem

6.2 Addition of a New FR to a Complete Design

One design scenario has been made to add one more FR to the original cam and pawl mechanism design problem. An FR, “increase stiffness,” is added, and a design solution, a DP, is searched in the database. By language matching, three kinds of ribs may be found with $FR_metric = 1.0$ from the database as described in the chapter 3. The three kinds of ribs can be automatically assembled to the pawl. In this case, GA generates combinations of two faces, which can be interfaced with the ribs. It generates 55 ($_{11}C_2$) combinations from the 11 faces of the pawl, and finds the best pair to give the best value of $INTERFACE_metric$ through the shape integration process. Figure 32 shows the pawl and cam mechanism and the three kinds of ribs to be interfaced together. Interfacing ways for all three kinds of the ribs will be the same. Thus, only triangular rib is shown in this example.

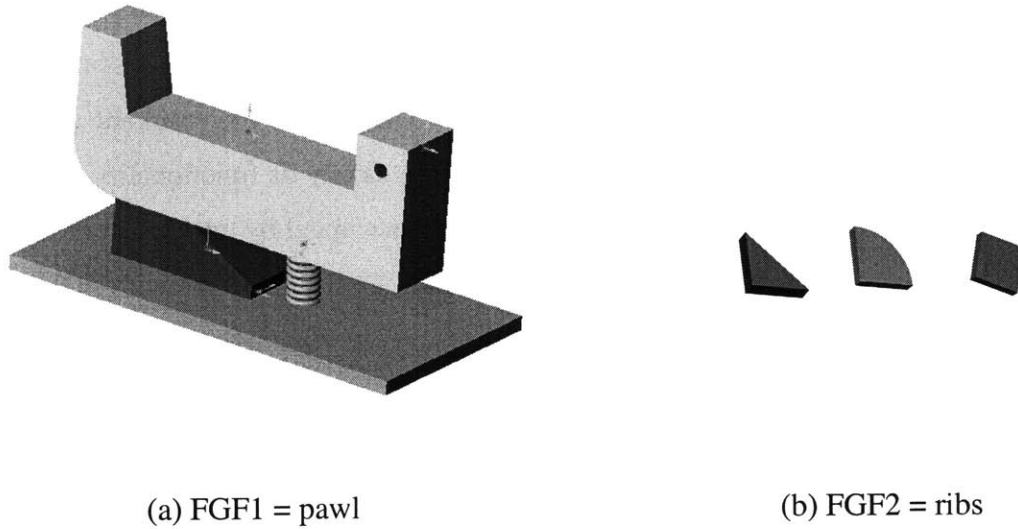


Figure 32. Addition of FGF2 to the pawl

Two methods, against:assembly ($T_T(3)$, $T_R(3)$) and against:assembly ($T_T(2)$, $T_R(1)$), are used to generate the assembled shapes and to calculate $INTERFACE_metric$ as follows.

Against:assembly ($T_T(3), T_R(3)$) (DOF, $\hat{N}_b^{Lb}, \hat{N}_i^{Li}, \vec{P}_b^{Lb}, \vec{P}_i^{Li}$) {

$[T_{Ri}^A] = \text{Orienting_against}(\hat{N}_b^{Lb}, \hat{N}_i^{Li});$

$[T_{Ti}^A] = \text{Positioning_against}(\hat{N}_b^{Lb}, \vec{P}_b^{Lb}, \vec{P}_i^{Li}, [T_{Ri}^A])$

$\vec{O}^B = \vec{P}_b^A, \hat{z}^B = \hat{N}_b^A$

Return(DOF($(\hat{x}^B, \hat{y}^B), (\hat{z}^B)$), $[T_{Ri}^A], [T_{Ti}^A], \vec{O}^B, \hat{z}^B$

}

Against:assembly ($T_T(2), T_R(1)$) (DOF, $[T_{Ri}^A], [T_{Ti}^A], \vec{O}^B, \hat{z}^B$) {

$\hat{y}^B = \frac{\hat{z}^B \times \hat{N}_b^A}{|\hat{z}^B \times \hat{N}_b^A|}, \hat{x}^B = \hat{y}^B \times \hat{z}^B$

$[T_{Ri}^B] = \text{Orienting_any}([T_C^B], \hat{N}_b^A, \hat{N}_i^A)$

AF_value = AF_metric($\hat{z}^B, \hat{N}_b^A, \hat{N}_i^A$)

$[T_{Ti}^B] = \text{Positioning_any}(\hat{x}^B, \vec{P}_b^A, \vec{P}_i^A, [T_C^B], [T_{Ri}^B])$

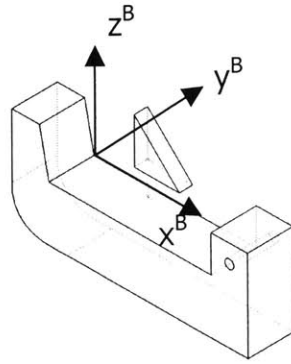
If($\hat{N}_i^A = \hat{z}^B$) $DOF_R = \hat{z}^B, DOF_T = \hat{x}^B, \hat{y}^B$

Else $DOF_R = 0, DOF_T = \hat{x}^B$

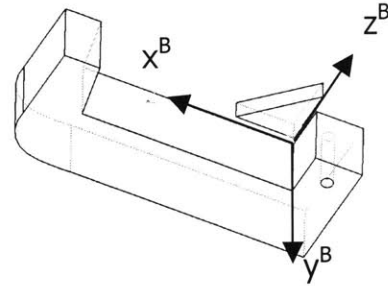
Return DOF(DOF_R, DOF_T), $[T_{Ri}^B], [T_{Ti}^B]$

}

Through the methods, the interfacing FGF, the rib, is oriented and positioned to the base FGF, the pawl, in the same way as described in section 5.1. Against:assembly ($T_T(3), T_R(3)$) outputs $T_T(2), T_R(1)$, and against:assembly ($T_T(2), T_R(1)$) outputs $T_T(1), T_R(0)$. A translational DOF, $T_T(1)$, is in the axis, Y^B , as shown in Figure 33. The INTERFACE_metric gives the highest value, 1.0, at the right concave corner of the pawl and the second highest value, 0.73 at the left concave corner of the pawl.



(a) Left concave edge



(b) Right concave edge

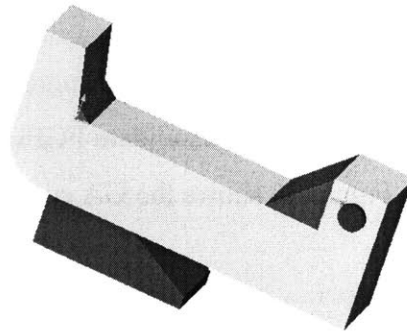
FR	DP	FR_metric	INTERFACE_metric
increase stiffness	triangular rib	1.0	Left corner: $1.0 \cdot (1 - 11.3^\circ / 360^\circ) \cdot (1 + 0.5) / 2 = 0.73$ Right corner: $1.0 \cdot 1.0 \cdot 1.0 = 1.0$

	DP42	DP11	DP12	DP2	DP31	DP32	DP41	DP5
FR42	X	-----	-----	-----	X			
FR11	-----	X			-----			
FR12	X	X	X		-----			
FR2	X		X	X	-----			
FR31	X	-----	-----	X	X	-----	-----	X
FR32	X			X	X	X		-----
FR41	X			X	X		X	-----
FR5			X		X	-----	-----	X

Figure 33. Results of interfacing a rib to the pawl

The size of DP5, the triangular rib, affects FR31. This coupling has been added on the design matrix constructed in the chapter 3. Thus, the size must be reduced not to affect the FR31. Figure 34 shows the resultant assembled shapes generated by the human designer based on the suggested locations of the FGF (triangular rib). The functional coupling described as a box on the design matrix in Figure 33 can be eliminated by reducing the size of the rib in the left

corner not to affect to FR31, “carry the force to the pin” by DP31 (a vertical surface of the pawl).



	DP42	DP11	DP12	DP2	DP31	DP32	DP41	DP5
FR42	X				X			
FR11		X						
FR12	X	X	X					
FR2	X		X	X				
FR31	X			X	X			
FR32	X			X	X	X		
FR41	X			X	X		X	
FR5			X		X			X

Figure 34. Resultant assembled shape for additional FR to pawl and cam design

6.3 Integration of FGFs for Beverage Can Design

This example shows the results of integration of more FGFs based on the beverage can example. From the following input FRs, four candidate FGFs are supposed to be found in the database as shown in Figure 35. Figure 36 shows the GA codes and one of GA chromosomes.

FR0 = contain and carry beverage

FR1 = enclose the bottom with resistance to an impact

FR2 = provide a volume to contain beverage

FR3 = reduce the material and increase stiffness of the body

FR4 = cover the body and provide pathway of beverage

DP0 = beverage can

DP1 = ? DP1 = bottom

DP2 = ? DP2 = hollow cylinder

DP3 = ? DP3 = conical section of the body

DP4 = ? DP4 = top

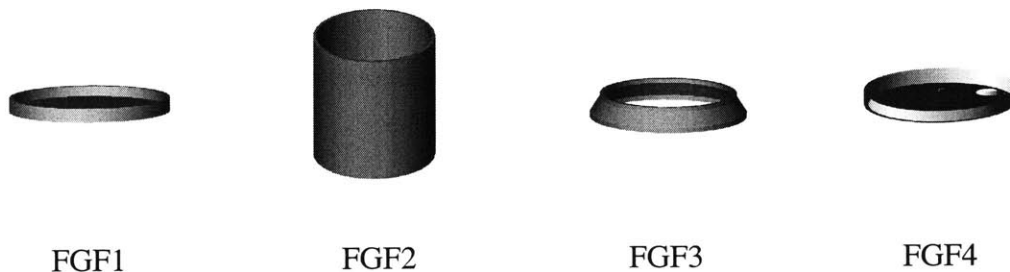


Figure 35. Definition of FRs and the corresponding candidate FGFs

FGF	Gene	Mating faces	FGF	Gene	Mating faces	FGF	Gene	Mating faces	FGF	Gene	Mating faces
	0										
1	1	A1	2	1	A2	3	1	A3	4	1	A4
				2	B2		2	B3			
				3	A2&B2		3	A3&B3			

(a) GA codes of mating faces

FGF1	FGF2	FGF2	FGF3	FGF3	FGF4	FGF4	FGF2	FGF4	FGF1	FGF3	FGF1
1	1	2	1	2	1	0	0	0	0	0	0

(b) Example chromosome

Figure 36. GA codes of mating faces and an example chromosome

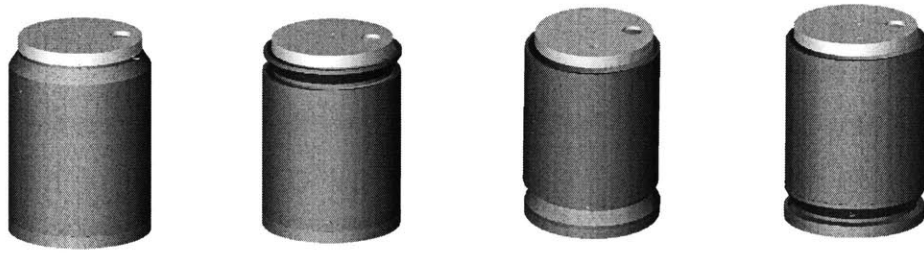
GA evolves to search the best matching of interface types to maximize fitness value defined as IT_metric . Then, the proposed assembly methods are applied to integrate FGFs into assembled shapes. In this example, GA evolves in 20 generations with 300 populations and only against:assembly ($T_T(3)$, $T_R(3)$) and against:assembly ($T_T(2)$, $T_R(1)$) are used. Figure 37 shows the resultant candidate shapes. (a) shows the assembled shapes after one more constraint, ‘concentric’. All the candidate shapes have $INTERFACE_metric = 1.0$. It means that all the candidate shapes are completely oriented and positioned satisfying all the pre-defined mating conditions between FGFs. (b) shows some other assembled shapes of $INTERFACE_metric = 0.389$. All these candidate shapes can be generated from three different chromosomes

FGF1	FGF2	FGF2	FGF3	FGF3	FGF4	FGF4	FGF2	FGF4	FGF1	FGF3	FGF1
0	0	1	1	0	0	0	0	1	1	0	0

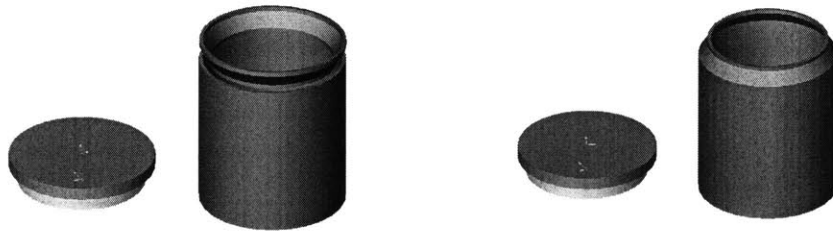
FGF1	FGF2	FGF2	FGF3	FGF3	FGF4	FGF4	FGF2	FGF4	FGF1	FGF3	FGF1
0	0	1	2	0	0	0	0	1	1	0	0

FGF1	FGF2	FGF2	FGF3	FGF3	FGF4	FGF4	FGF2	FGF4	FGF1	FGF3	FGF1
0	0	3	3	0	0	0	0	1	1	0	0

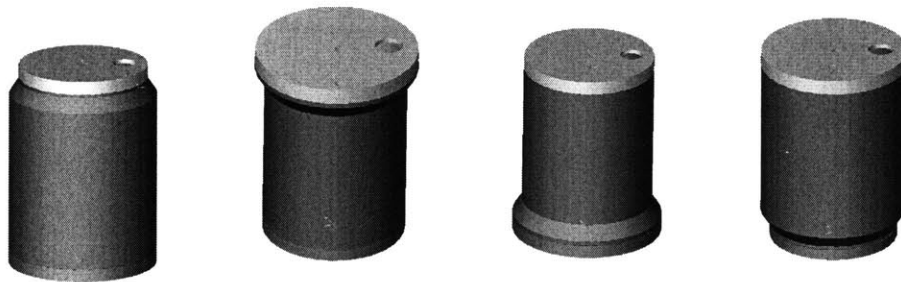
In the first chromosome, one pair of mating faces has been missed to be interfaced among total three pairs of mating faces. Thus, IT_metric is 0.67 ($= 2*2/6$). However, all the mating faces given by the chromosome can be correctly oriented and positioned: AF_metric = 1.0 and Pos_metric = 1.0. The first chromosome represents the first candidate shape in (b) and has INTERFACE_metric = 0.67 (IT_metric*AF_metric*Pos_metric). The same analysis can be applied to the second chromosome that represents the second candidate shape in (b). The third chromosome represents interfaces of all the pairs of mating faces. In this case, IT_metric is 1.0 ($= 3*3/6$), because all the interface types (against:assembly) can be matched correctly. Also, all the mating faces can be oriented correctly so that AF_metric = 1.0. However, one pair of mating faces cannot be positioned geometrically. This fact gives Pos_metric = 0.583 ($((1+(1+0.5)/2)/3)$). INTERFACE_metric for the third chromosome is 0.583 (IT_metric*AF_metric*Pos_metric). There are two combinations for interfacing two pairs of mating faces; A2:A3 -> B2:B3 and A2:B3 -> B2:A3. These two combinations show two different candidate shapes in (b). The value of INTERFACE_metric in this case highly depends on a value, which is pre-set to 0.5 in Pos_metric. The pre-set value is used for degrading INTERFACE_metric if positioning is failed. If we set the value to a larger, INTERFACE_metric can be increased for the same case.



(a) Assembled shapes with $\text{INTERFACE_metric} = 1.0((2*3/6)*(3/3)*(3/3))$



(b) Assembled shapes with $\text{INTERFACE_metric} = 0.67((2*2/6)*(1/1)*(1+1)/2)$ or $0.583((2*3/6)*(1/1)*(1+(1+0.5)/2)/3)$



(c) Shapes through scaling with $\text{INTERFACE_metric} = 1.0$

Figure 37. Resultant shapes for the beverage can design problem

Figure 38 shows the advantage of top-down decompositions for combining FGFs. The body of the beverage can is assembled first, and then the bottom, the assembled body, and the top are assembled. In this case, the length of chromosome for assembling the body is just 2 and that for the next assembly is 6. Each length is shorter than 12, which is the length for assembling all four FGFs from the same level. Actual GA evolution of the two step assembling is 3 generations with 200 populations plus 10 generations with 300 populations to get the same results as shown in Figure 37. Total calculations are 3600 chromosomes for this case. Thus, it is less than 6000 calculations in the other case, which has 12 length chromosomes. This is rough estimation for GA calculations, but explains the advantage of top-down decompositions for bottom-up integrations.

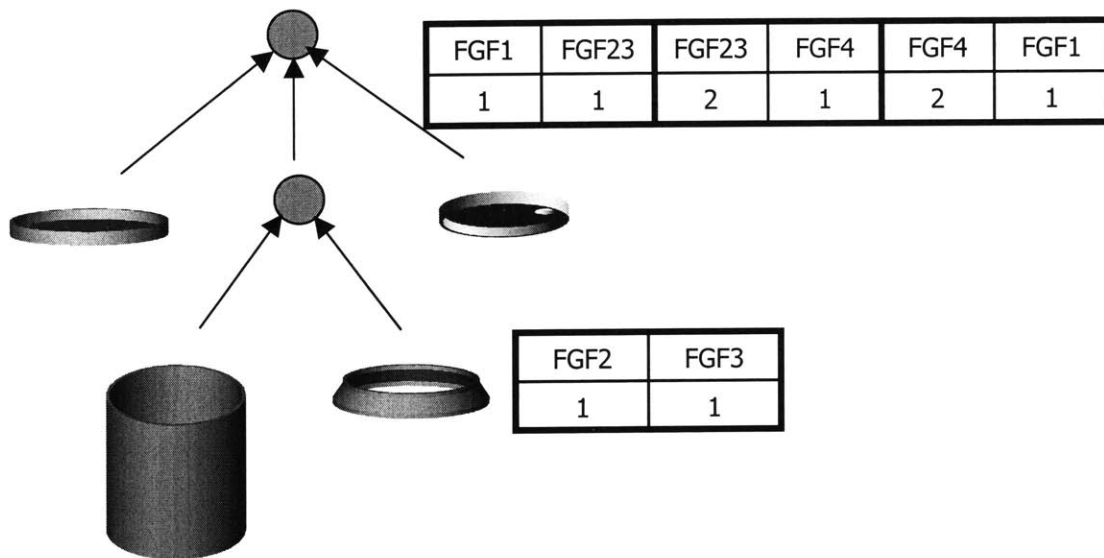


Figure 38. Advantage of top-down decomposition

6.4 Inference of FRs from Interfaces

In some cases, human designer has ignorance of FRs, what he/she wants to achieve, due to the lack of knowledge. This example shows how the method proposed in this thesis can help search the related design elements by geometric information. If a designer knows one

FR, “transform gas pressure into translational motion,” and the database provides a DP, “piston,” and the corresponding FGF as a candidate design solution. The corresponding FGF has three dangling interfaces; ‘against:assembly,’ ‘against:assembly,’ and ‘fits:assembly:cylinder’. The dangling interfaces mean that the piston has been interfaced with another FGF in the design case. Computer algorithm can search FGFs with three interfaces that are matched with the dangling interfaces of the piston. It is supposed that three candidate FGFs be found in the database and the designer can select the most proper one. Figure 39 shows this process. The three FGFs, ‘piston ring,’ ‘washer,’ and ‘supporter’ may have been found and ‘piston ring’ may be selected to be interfaced with the piston. The additional FRs and DPs can be inferred by this process. Here, ‘piston ring’ and ‘supporter’ have the same topology. Thus, the dimensions of them have important roles to satisfy two different FRs by different shapes.

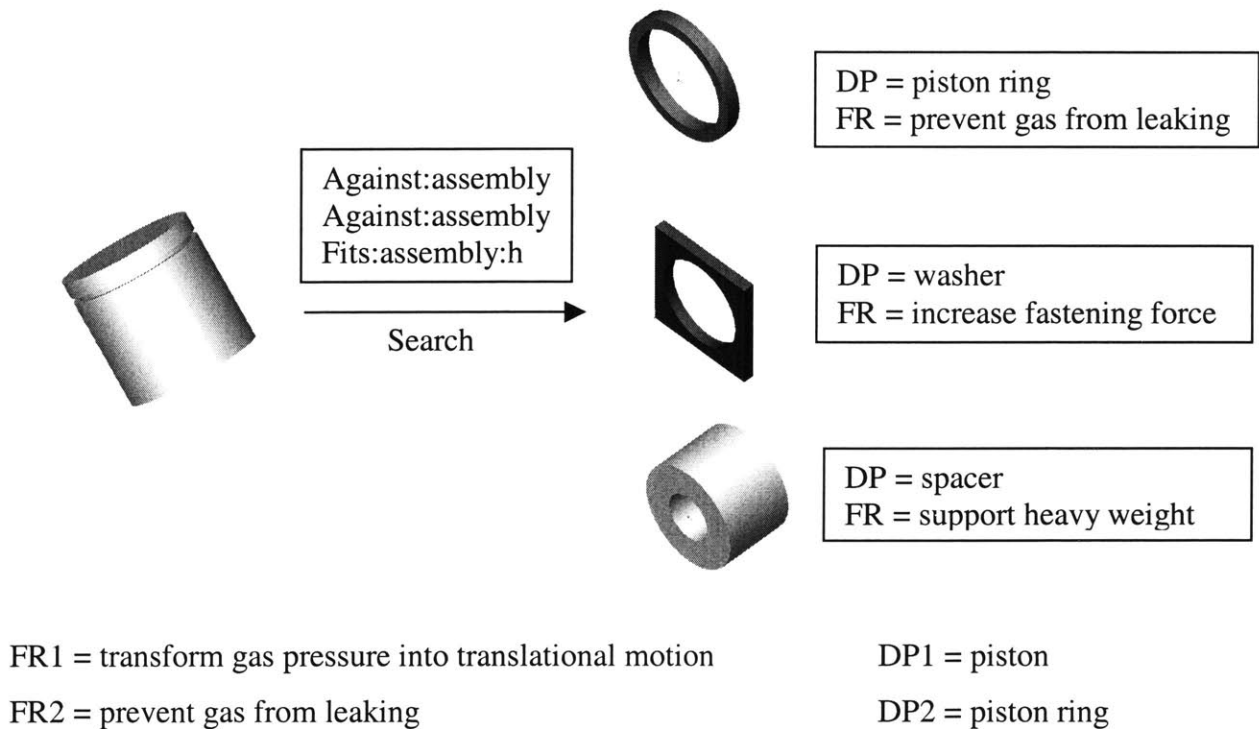


Figure 39. Example for inferring FR and DP from interfaces

Chapter 7

Conclusions

7.1 Concluding Remarks

This thesis presented a method that automatically generates assembled shapes from input functional descriptions. The method includes V-model design process, database, and computer algorithms for searching candidate FGFs and automatically assembling them. The method generated interesting and reasonable shapes, which are satisfying input FRs. FR_metric (0.0 ~ 1.0) and INTERFACE_metric (0.0 ~ 1.0) have been made to rank the generated shapes and to support decision-making by computer algorithms and/or human designers. FR_metric is a language-matching rate between an input FR statement and a candidate FR statement. INTERFACE_metric (0.0 ~ 1.0) is an interface-matching rate (interface type and geometric assemble-ability) between candidate FGFs. The computer algorithms can filter a lot of the generated shapes by the values of FR_metric first, and then by the values of INTERFACE_metric. The candidate shapes with FR_metric = 1.0 and INTERFACE_metric = 1.0 can be considered as good design solutions among a lot of generated shapes in this thesis. Design matrix shown in the chapter 2 and 3 must be constructed for each candidate shape to check satisfaction of all the FRs and finally to determine the best design solution.

This sort of shape generation using database is a combinatorial problem with a lot of combinations and high computational complexity. This thesis used several important techniques to handle the heavy computational load in a manageable level.

- All the detailed geometry has been encapsulated. The developed computer algorithms use only geometric information on interfaces, i.e. mating faces. Thus, the number of combinations and computational complexity are relatively small comparing to other approaches using complex whole geometry.
- Several modules of the computer algorithms have been used. The language-matching module reduces the solution space a lot in a certain level by FR_metric. The module of GA evolution reduces the solution space again by matching pre-defined interface-types between FGFs by IT_metric. Then, automatic assembling algorithm operates assembling of FGFs and calculates values of INTERFACE_metric. Because solution spaces are reduced through the steps between the algorithms, computational complexity of automatic assembling algorithm is in a manageable level.
- Bottom-up integrations of FGFs in top-down hierarchical tree can eliminate many combinations and reduce search space for allowable combinations comparing to integrations only by bottom-up manner.

7.2 Contributions

The followings are discussions about the contributions that have been made.

Design Process for Intelligent CAD system

The V-model design process is an important basis for intelligent CAD systems for conceptual design. It combines the advantages of top-down decomposition for searching proper information and of bottom-up approach to integrate the lower level geometric entities to satisfy higher level FRs. This is because the zigzagging decomposition between the functional domain and the physical domain, as suggested in the axiomatic design theory, produces well-defined hierarchical trees of FRs, DPs and geometric entities. The lower level

DPs must be subsets or elements of a higher level DP in the hierarchical tree by these logical decompositions. The existing CAD programs generate solid parts first and then interface them into an assembly without explicitly representing functional aspects of the generated shapes in the design process. The most CAD programs represent a shape by a decomposed feature tree generated by solid modeling technique only in the physical domain, but the V-model starts to decompose a design problem in the functional domain and finds solutions in the physical domain. This design process generates relations between functional descriptions and geometric entities in hierarchical trees. The incorporation of the functional aspects to the CAD models in a systematic way based on a thinking process – the essence of the V-model – in conceptual design stage is an important step in computer-aided design. A CAD system implemented by the V-model design process presented in this thesis will enhance understandability and reusability of design concepts related to the CAD models, and increase degrees of freedom for creative design.

Knowledge-based system, Case-based system, and Intelligent CAD system

Most systems to support conceptual design have been focusing on searching and relating information stored in the database to support search of design solutions. For example of Invention Machine [35], Concept Database [36], Design Components [27] and most case-based design systems, technology of searching and relating information makes us to find relevant concepts and visualizes them. A lot of progress has been made to generate language explanations on the design solutions by composing lexical elements from many design concepts, but research on generating integrated shapes or pictures from the pieces of geometric elements or images is rare, because it is difficult to automate the integration. In most systems, a lot of drawings or pictures are popped up from input queries by language processing or inference technique. [37] In general, human designers analyze the drawings or the pictures, separate or collect the important geometric or image elements, and finally regenerate the integrated candidate shapes or images using sketch tool or CAD software. This process is very time consuming or sometimes not tractable by human designers, if the number of extracted drawings or pictures is large. Instead, our method can automatically generate

assembled shapes as the design solutions for conceptual design, which are not just a collection of drawings or pictures.

Feature based design

A concept of functional geometric feature (FGF) has been defined for feature based design systems. If any geometric entities can be separated into geometric volumes, called cells, from the whole solids based on FRs, it can be a big progress to reuse them by functional purposes in feature based design systems. Conceptually, the V-model makes this possible. In most research on feature-based design, geometric features are highly related to solid modeling techniques (extrusion, cut, hole, round, and so on) or manufacturing features (slots, pockets, steps, and so on). In contrast, our effort has been made to relate geometric features to functional descriptions by using the V-model.

There may exist some cases, in which geometric entities cannot be separated from a whole solid. In those cases, we propose that the corresponding geometric features do not have to be separated. The cell that represents more than one FGFs can be defined to satisfy the corresponding FRs. The importance of the V-model resides on that it can make links of each geometric features to corresponding FRs in a logical and systematic way. In some other cases, one FGF may be redundantly used to integrate in a level of a hierarchical tree. We limits redundant use of one FGF only with redundant definition of one FR in this thesis

There must exist more than one ways of decomposing a shape into FGFs. Because of this reason, it is not easy to automate the shape decomposition process. To overcome this obstacle, we did not make an effort to find out a general way to decompose shapes to FGFs. Instead, we allow different ways of decomposition of a shape as different cases. The automatic shape generation algorithm can combine all the cases.

7.3 Vision and Future Work

The method presented in this thesis has demonstrated that a "functional CAD" can aid designers in generating conceptual design solutions from functional descriptions, in reusing existing CAD models, and in creating new designs. Our vision in the future is to make large distributed databases of FGFs, generated from a lot of existing CAD models distributed over the world, and to use the databases for automatic candidate shape generation for conceptual design from functional descriptions through the Internet. To achieve this vision, the following topics should be covered further.

- Interference checking between FGFs
- More types of assembly interfaces
- Shape deformation and synthesis for rigid attachment
- Data transportation techniques between large distributed databases

References

- [1] M. J. Pratt and P. H. Wilson, "Requirements of Support of Form Features in a Solid Modeling System," CAM-I Report, R-85-ASPP-01, Arlington, Texas, 1985
- [2] J. R. Dixon and J. J. Cunningham, "Research in Designing With Features," Proceedings of the IFIP TC 5/WG 5.2 Workshop on Intelligent CAD, edited by H. Yoshikawa and D. Gossard, pp. 137 – 148, Boston, MA. USA. 1987
- [3] P. C. Sreevalsan and J. J. Shah, "Unification of Form Feature Definition Method," Proceedings of the IFIP WG 5.2 Working Conference on Intelligent Computer Aided Design, edited by D. C. Brown, M. B. Waldron, and H. Yoshikawa, pp. 83 – 99, Columbus, OH, USA, 1992
- [4] N. P. Suh, "Principles of Design," Oxford University Press, 1990
- [5] N. P. Suh, "Axiomatic Design: Advances and Applications," Oxford University Press, 2000
- [6] D. Xue, H. Takeda, and H. Yoshikawa, "An Intelligent Integrated Interactive CAD – A Preliminary Report," Proceedings of the IFIP WG 5.2 Working Conference on Intelligent Computer Aided Design, edited by D. C. Brown, M. B. Waldron, and H. Yoshikawa, pp. 163 – 187, Columbus, OH, USA, 1992
- [7] A. Aamodt and E. Plaza, "Case-based Reasoning: Foundational Issues, Methodological Variations, and System Approaches," AI Communications, IOS Press, Vol. 7:1, pp. 39 – 59, 1994
- [8] D. B. Lenat, "The Dimension of Context Space," CYC Corporation, 1998
- [9] H. Sakurai and D. C. Gossard, "Recognizing Shape Features in Solid Models," IEEE Computer Graphics and Applications, v. 10, no. 5, pp. 22 – 32, September, 1990
- [10] H. Sakurai, "Volume Decomposition and Feature Recognition: Part I – Polyhedral Objects," Computer Aided Design, Vol. 27, Issue 11, pp. 833 – 843, 1995
- [11] E. Wang and Y. S. Kim, "Form Feature Recognition Using Convex Decomposition: Results Presented at the 1997 ASME CIE Feature Panel Session," Computer Aided Design, Vol. 30, Issue 13, pp. 983 – 989, 1998

- [12] B. Li and J. Liu, "Detail Feature Recognition and Decomposition in Solid Model," *Computer Aided Design*, Vol. 34, pp. 405 – 414, 2002
- [13] Y. Woo, "Fast Cell-based Decomposition and Applications to Solid Modeling," *Computer Aided Design*, accepted on August 1, 2002
- [14] Y. Lu, R. Gadh, and T. J. Tautges, "Feature Based Hex Meshing Methodology: Feature Recognition and Volume Decomposition," *Computer Aided Design*, Vol. 33, Issue 3, pp. 221 – 232, 2001
- [15] A. Dong and A. M. Agogino, "Text Analysis for Constructing Design Representations," *Artificial Intelligence in Engineering*, Vol. 11, pp. 65 – 75, 1997
- [16] P. J. Bentley, "Genetic Evolutionary Design of Solid Objects using a Genetic Algorithm," PhD Thesis, University of Huddersfield, U.K., 1996
- [17] M. Peabody and W. C. Regli, "Clustering Techniques for Databases of CAD Models," Technical Report DU-MCS-01-01, Drexel University, PA., 2001
- [18] E. Charniak and D. McDermott, "Introduction to Artificial Intelligence," Addison-Wesley Publishing Company, 1985
- [19] R. A. Brooks, C. Breazeal, R. Irie, C. C. Kemp, M. Marjanovi_c, B. Scassellati, M. M. Williamson, "The Cog Project: Building a Humanoid Robot," 1998
- [20] Lipson, H., Pollack J. B., 2000, "Automatic Design and Manufacture of Artificial Lifeforms," *Nature* 406, pp. 974-978
- [21] S. Do and N. P. Suh, "Object-Oriented Software Design with Axiomatic Design," *Proceedings of International Conference on Axiomatic Design*, 2000
- [22] T. Taura and H. Yoshikawa, "A Metric Space for Intelligent CAD," *Proceedings of the IFIP WG 5.2 Working Conference on Intelligent Computer Aided Design*, edited by D. C. Brown, M. B. Waldron, and H. Yoshikawa, pp. 133 – 157, Columbus, OH, USA, 1992
- [23] H. Yoshikawa, "General Design Theory and a CAD System," *Man-Machine Communication in CAD/CAM*, *Proceedings of IFIG Working Conference*, Tokyo, Japan, 1982
- [24] Y. Umeda, M. Ishii, M. Yoshioka, Y. Shimomura, and T. Tomiyama, " Supporting Conceptual Design based on the Function- Behavior- State Modeler," *Artificial*

- Intelligence for Engineering Design, Analysis and Manufacturing, Vol. 10, pp. 275 – 288, 1996
- [25] J. S. Gero, K. W. Tham, and H. S. Lee, “Behavior: A Link between Function and Structure in Design,” Proceedings of the IFIP WG 5.2 Working Conference on Intelligent Computer Aided Design, edited by D. C. Brown, M. B. Waldron, and H. Yoshikawa, pp. 193 – 220, Columbus, OH, USA, 1992
 - [26] N. P. Suh and S. Sekimoto, “Design of Thinking Design Machine,” CIRP Annals, 1990
 - [27] P. Clark and B. Porter. Building Concept Representations from Reusable Components. AAAI'97, pp. 369-376, CA:AAAI Press, 1997
 - [28] H. Toriya and H. Chiyokura, “3D CAD: Principles and Applications,” Springer-Verlag, 1991
 - [29] M. El-Mehalawi and R. A. Miller, “A Database System of Mechanical Components based on Geometric and Topological Similarity. Part I: Representation,” Computer Aided Design, Vol. 35, Issue 1, pp. 83 – 94, 2003
 - [30] K. Lee and D. C. Gossard, “A Hierarchical Data Structure for Representing Assemblies: Part 1,” Computer Aided Design, Vol. 17, No. 1, pp15 – 19, 1985
 - [31] K. Lee and G. Andrews, “Inference of the positions of components in an assembly: Part 2,” Computer Aided Design, Vol. 17, No. 1, pp20 – 23, 1985
 - [32] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller, “Introduction to WordNet : An On-line Lexical Database,” <http://www.cogsci.princeton.edu/~wn>, August, 1993
 - [33] R. Gold and A. Rangarajan, “A Graduated Assignment Algorithm for Graph Matching,” IEEE Trans Pattern Anal Mach Intell, 18(4), pp. 377 – 388, 1996
 - [34] R. P. Paul, “Robot Manipulators: Mathematics, Programming and Control,” The MIT Press, 1984
 - [35] <http://www.invention-machine.com>
 - [36] http://best.me.berkeley.edu/cdb_home
 - [37] R. D. Coyne and J. S. Gero, “Knowledge-Based Design Systems,” Addison-Wesley, 1990