UNIVERSITÉ DE MONTRÉAL

AN APPROACH FOR ASSEMBLY SEQUENCE PLANNING USING SOLID MODELS

B. REDDY GOTTIPOLU

DÉPARTMENT DE GÉNIE ÉLECTRIQUE ET DE GÉNIE INFORMATIQUE

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION

DU DIPLÔME DE PHILOSOPHIAE DOCTOR (Ph.D.)

(GÉNIE ÉLECTRIQUE)

NOVEMBRE 1994

ISBN 0-315-99733-8

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée:

AN APPROACH FOR ASSEMBLY SEQUENCE PLANNING USING SOLID MODELS

présentée par: GOTTIPOLU B. Reddy

en vue de l'obtention du diplôme de: Philosophiae Doctor (Ph.D.)

a été dûment acceptée par le jury d'examen constitué de:

To My Parents

SOMMAIRE

L'assemblage est habituellement une opération importante en termes de main-d'oeuvre dans plusieurs types d'industries puisque cette fonction contribue significativement au coût du produit manufacturé. Ainsi, une planification efficace des activités d'assemblage peut réduire considérablement le coût d'assemblage et diminuer les temps de cycle de fabrication. Donc, la détermination d'une bonne séquence d'assemblage apparaît comme une étape cruciale de la planification d'assemblage. D'ailleurs, le choix d'une bonne séquence d'assemblage joue un rôle extrêmement important lors de la sélection des caractéristiques du procédé d'assemblage et du produit fini. Le développement d'un système intégré à l'étape de la conception devient nécessaire en raison de l'utilisation croissante des modèles géométriques dans la conception de produits.

Dans le cadre de cette recherche, il est question d'une brève revue des méthodes de détermination des séquences d'assemblage, leur applicabilité, leurs avantages et leurs inconvénients. Les sujets qui doivent être couverts par une planification intégrée d'assemblage sont la modélisation d'assemblage, la représentation des connaissances de base et ce que l'on peut en tirer, la génération de séquences d'assemblage, la représentation des séquences et enfin l'analyse de ces séquences pour retenir la meilleure. L'étude suggère un Système de Planification Intégrée d'Assemblage assisté par Ordinateur (SPIAO, CIAPS - Computer Integrated Assembly Planning System) qui intègre la conception et la planification d'assemblages mécaniques. Pour planifier les séquences d'assemblage, l'approache developée se compose en trois phases. Lors de la premiére phase, il s'agit d'abord de créer des modèles solides d'assemblage en utilisant le logiciel PADL-2. On fait appel à un algorithme structuré à détection de collisions de piéces de façon à générer les relations de précédences entre les composants de l'assemblage. La connaissance des priorités est déduite à partir des fonctions de contact (C) et de translation (T) pour chaque paire de composants. Cette information est retenue sous la forme d'un modèle relationnel et ce modèle est assez flexible pour y inclure toute information additionnelle.

Lors de la seconde phase, l'algorithme de génération des séquences d'assemblage prend connaissance des informations sur les contacts et les précédences à partir du modèle relationnel, puis génère toute les séquences géométriquement réalisables. Cet algorithme traduit d'abord les fonctions C et T en tables de vérités, puis les principes d'algèbre Booléen sont par la suite appliqués afin de vérifier la faisabilité des assemblages. Deux procédures en découlent. La première répertorie toutes les paires d'assemblage réalisables en vérifiant la fonction C; la seconde procédure génère des sous-assemblages d'ordre supérieur par l'analyse des deux fonctions C et T pour toutes les paires considérées. Les séquences générées sont alors schématisées selon deux formats: un graphique des séquences d'assemblage (ASG, Assembly Sequence Graph) et une table de séquences d'assemblage (AST, Assembly Sequence Table).

La troisième phase traite de la sélection et de l'évaluation des séquences précédemment générées. On étabit plusieurs contraintes stratégiques de même que des considérations rencontrées lors du choix et l'évaluation de séquences d'assemblage. L'imposition de tels critères réduit le nombre de séquences possibles à un niveau pratique. Des moyens sont fourni qui permettent d'éliminer les états ou les tâches d'assemblage indésirés des graphiques et des tables des séquences d'assemblage. Les séquences retenues sont par la suite soumises à une évaluation quantitatives avec le logiciel DFA (Design for Assembly). Ce logiciel fournit une planification d'assemblage optimale pour une utilisation pratique compte tenu des ressources disponibles. On démontre ensuite la validité et l'applicabilité des approches proposées en les illustrant d'exemples pratiques tirés de l'expérience du manufacturier d'appareils électroménagers Frigidaire Canada.

L'approache qui est proposée dans cette recherche puise toute les informations directement d'un modèle de CAO (Conception Assistée par Ordinateur) pour générer des assemblages et ce système permet aussi l'assemblage de composants selon une multitude de directions. Par ailleurs, cette approche peut être considérée comme étant un outil important à l'ingénieur de fabrication lors de l'implantation de concepts d'ingénierie simultanée.

ABSTRACT

Assembly is usually a labor intensive operation and in many types of industries, it contributes significantly to the product cost. Consequently, there is a lot of interest in reducing the cost attributed to assembly activities. An efficient assembly planning approach can reduce the assembly cost and the manufacturing lead time considerably. A crucial step in the assembly planning is the determination of the assembly sequence, which plays an important role in determining the characteristics of the assembly process as well as the finished assembly. The development of an integrated system from the design phase has become pertinent because of the growing use of geometric modeling in the design of products.

In this research, a brief review of assembly sequence determination methods, their applicability, and merits and demerits is provided. The research issues that must be addressed in integrated assembly planning are identified as assembly modeling, precedence knowledge extraction and representation, generation of assembly sequences, representation of sequences and analysis of sequences to select the best sequence. A computer integrated assembly planning system (CIAPS) that integrates design and planning of mechanical assemblies has been proposed. The developed approach implements a three phase assembly sequence planning to address these issues. It starts with the creation of solid models of the assembly using the PADL-2 solid modeling package. A robust part collision detection algorithm to generate precedence relationships among the components of assembly is included in the system. The precedence knowledge is extracted in terms of contact (C) and translational (T) functions for each pair of components. This information is stored in the form of a relational model and this model is flexible enough to include any additional information.

In the second phase, an assembly sequence generation algorithm takes contact and precedence information from the relational model as input and generates all the geometrically feasible sequences. This algorithm first translates the C and T functions into truth tables, and Boolean algebra principles are applied to verify the feasibility of assembly. It consists of two procedures. The first procedure lists all the feasible assembly pairs by checking the C function and the second procedure generates higher order subassemblies by analyzing both C and T functions of all the involved pairs. The generated sequences are then represented by two representation schemes: Assembly sequence graph (ASG) and Assembly sequence table (AST).

The third phase deals with the selection and evaluation of the generated sequences. Various strategic constraints and considerations encountered in the selection and evaluation of assembly sequences are addressed. The number of feasible sequences is reduced to a practical level by imposing these criteria. Editing features are provided to eliminate the unwanted assembly states or assembly tasks from the assembly sequence graph and assembly sequence table. The selected candidate sequences are then subjected to quantitative evaluation using the DFA (Design for Assembly) Toolkit software to obtain an optimal assembly plan for practical use within the available resources. The validity and applicability of the proposed approach are demonstrated using two practical examples, one of them selected from Frigidaire Canada, a home appliances manufacturer.

The approach proposed in this research extracts all reasoning information directly from the CAD model of the assembly and permits the assembly of components in a multitude of directions. This can be considered as an important aid to the manufacturing engineer in implementing concurrent engineering concepts.

RÉSUMÉ

L'assemblage est un des importants procédés manufacturiers dans plusieurs sortes d'industries, incluant l'importante industrie automobile et celle des biens électroniques de consommation. Ce domaine est devenu traditionnellement un secteur où les coûts directs de main-d'oeuvre sont appréciables et en conséquence les temps de fabrication de ces produits sont accrus. De plus, dans plusieurs secteurs de l'industrie, l'assemblage contribue significativement à accroître le coût du produit et parfois même, cette activité compte pour près de quarante pourcent du coût total de fabrication. En conséquence, il devient très intéressant de pouvoir réduire les coûts d'assemblage. Une approche par planification peut largement réduire les coûts de fabrication et ainsi augmenter la productivité. Il est de même pour la qualité du produit. La tendance actuelle qui consite à changer constamment la conception du produit et les stratégies de fabrication qui s'y rattachent démontrent la très grande importance des systèmes de planification automatisés. Qui plus est, l'usage croissant des ordinateurs et des systèmes à modélisation géométrique pour la conception d'un produit et de son assemblage conduisent au développement de systèmes de planification intégrés et ce, de la phase de conception jusqu'à l'étape finale d'assemblage.

La planification de procédés assistée par ordinateur (PPAO, CAPP - Computer-Aided Process Planning) pour l'assemblage fait appel dès les débuts de la conception à la préparation d'un plan détaillé de l'assemblage du produit. L'information de base contenue dans un plan d'assemblage est la séquence d'assemblage, qui spécifie l'ordre dans lequel les composants peuvent être assemblés afin de générer l'assemblage final. La planification de la séquence d'assemblage quant à elle, peut prendre la forme d'un

problème scientifique générique selon lequel les propriétés fondamentales des relations d'assemblage, les géométries et les opérations sont utilisées comme guide pour la recherche d'une séquence acceptable, complète, optimale ou même encore une séquence recherchée.

La détermination d'une séquence d'assemblage appropriée constitue une étape critique lors de l'élaboration d'un système d'assemblage. De plus, la séquence choisie affecte plusieurs aspects du procédé d'assemblage, telle que la définition des sous-assemblages, la sélection des équipements du procédé, la conception d'outils spéciaux, etc. Le choix d'une séquence, selon laquelle les pièces sont assemblées de façon à former un assemblage, joue un rôle primordial lorsqu'il s'agit de déterminer les caractéristiques des tâches d'assemblage et d'assemblage final.

Un produit typique peut avoir plusieurs centaines à plusieurs milliers de séquences d'assemblage et ce nombre croît exponentiellement avec le nombre de composants qui forment l'assemblage. Le choix d'une séquence à partir d'une panoplie d'alternatives constitue une étape cruciale si l'on souhaite spécifier les cartactéristiques des tâches de l'assemblage comme par exemple les difficultés de l'opération assemblage, les besoins en fixation, les posssibilités qu'une pièce soit endommagée lors du procédé d'assemblage, le coût d'assemblage, etc. Donc il est essentiel pour l'ingénieur de fabrication de déterminer une séquence d'assemblage satisfaisante parmi toutes les solutions permises eu égard aux ressources disponibles.

Malheureusement, trop peu d'aides à la planification existent pour l'analyse d'un assemblage en raison de la nature subjective et indéfinie de la génération des séquences d'assemblage et du choix du procédé. Jusqu'à maintenant, la génération d'une séquence d'assemblage demeure une démarche sans structure et requiert énormément d'expertise et de connaissances du génie manufacturier. L'objectif principal de cette recherche est de développer un système intégré de planification d'assemblage qui puisse choisir une séquence optimale directement à partir d'un modèle solide tridimensionnel devant satisfaire des contraintes de priorité et d'autres contraintes spécifiées par l'utilisateur.

Les sujets de l'heure en recherche sur la planification d'assemblage automatisé sont les suivants:

- Modélisation ou représentation d'assemblages pour décrire la géométrie, la topologie des composants ainsi que les relations entre les pièces, etc.
- L'identification, l'acquisition et la représentatiopn de relations de précédences nécessaires pour éviter l'interférence géométrique lors de l'assemblage.
- La génération de toutes les séquences d'assemblage réalisables qui satisfont les relations de précédences.
- La représentation efficace des plans d'assemblage générés pour analyses ultérieures.
- La détermination de séquences d'assemblage optimales minimisant les temps et les coûts d'assemblage étant donné un environnement spécifique et des facteurs stratégiques connus.

Dans cette thèse, il sera question d'une brève revue des différentes méthodes de détermination des séquences, comment ces méthodes ont été élaborées en fonction des thèmes énumérés ci-haut, leurs avantages et leurs inconvénients. En raison de la complexité du problème tel qu'expliqué précédemment, il est pratiquement impossible d'utiliser des méthodes de planification d'assemblage existantes pour déterminer une séquence pour traiter des assemblages contenant un grand nombre de pièces.

En gardant l'objectif de développer un système de planification de séquences d'assemblage, il sera question de déterminer directement une séquence d'assemblage optimale et réalisable à partir d'un modèle d'assemblage tridimensionnel (déterminé par CAO). Tout au long de cette recherche, nous poursuivrons ce but en suivant trois phases. La méthode débute avec la création d'une représentation CSG de chacun des composants de l'assemblage selon leur position au niveau de l'assemblage et ce, en faisant appel à un système de modélisation solide PADL-2. L'information relationnelle et les connaissances de priorités sont représentées sous forme de modèle relationnel. Ce dernier est formé principalement de deux fonctions binaires 1x6: plus précisément une fonction de Contact (C-function) et une fonction de translation (T-function) pour chaque paire de composants de l'assemblage. La fonction C apporte l'information concernant la présence ou l'absence de contact entre une paire de composants tandis que la fonction T fournit l'information sur les trajectoires sans collision lorsque les composants sont séparés. Les fonction C et T sont automatiquement extraites d'un modèle d'assemblage géométrique et se fondent sur une procédure de détection de collisions pour chaque paire de composants. Une des plus importantes possibilités permise par cette approche est l'aptitude à transformer une description tridimensionnelle d'un assemblage en contraintes de mouvements.

Au cours de la seconde phase, l'étude propose un algorithme de génération de séquences d'assemblage. L'algorithme génère toutes les séquences géométriquement réalisables compte tenu des rigides contraintes imposées par la géométrie et la topologie des composants individuels de même que par l'assemblage entier. Cet algorithme traduit d'abord les fonctions C et T en tables de vérités et ensuite on utilise les principes de l'algèbre Booléen pour vérifier la faisabilité de l'assemblage. Cette étape se divise en deux procédures. La première liste toutes les paires d'assemblage réalisables en vérifiant la fonction C. A partir de deux sous-assemblages, la seconde procédure vérifie la possibilité d'ajouter un composant ou un autre sous-assemblage au sous-assemblage déjà existant. Cette étape est répétée jusqu'à obtention d'un assemblage final. Les séquences générées sont alors représentées d'une part graphiquement grâce à la méthode ASG (Assembly Sequence Graph), puis d'autre part sous forme de table par la méthode AST (Assembly Sequence Table). Ces deux représentations sont tout à fait complètes puisqu'elles permettent d'évaluer toutes les séquences et permettent ultérieurement de les analyser.

La troisième phase traite de la sélection et de l'évaluation des séquences générées. Plusieurs contraintes non géométriques ou stratégiques (contraintes souples) doivent être considérées lors du choix et l'évaluation de ces séquences. En imposant ces critères, le nombre de séquences réalisables peut être réduit à une limite pratique. Quelques améliorations présentées dans ce document permettent d'utiliser la méthode ASG ou AST de façon à éliminer les états ou les tâches d'assemblage indésirables. Les critères qui contraignent les séquences d'assemblage décrites dans cette recherche influencent significativement le coût unitaire et la configuration du système d'assemblage. Ainsi, les séquences retenues deviennent sujettes à l'analyse quantitative

par l'utilisation du logiciel DFA (Design for Assembly). Ce faisant, on obtient une planification optimale de l'assemblage qui plus est, offre une solution d'ordre pratique eu égard aux ressources disponibles. Cette phase, au cours de laquelle on considère des contraintes imposées par des propriétés non géométrique associées à la conception du produit et aussi à l'aisance de l'assemblage, avance la planification à un niveau tel que l'on peut pratiquement utiliser cette planification pour résoudre des problèmes réels qui surgissent dans la vie industrielle. La représentation des séquences est assez bien structurée et elle peut mener à l'élaboration de plusieurs critères qualitatifs et quantitatifs.

La complexité des calculs qu'occasionne l'emploi d'un algorithme de génération géométrique de séquences faisables a été évalué en déterminant le nombre de combinaisons à analyser. Ce nombre dépend non seulement du nombre de pièces à assembler, mais aussi de la façon avec laquelle on peut assembler ces pièces. Les mesures de cette complexité sont estimées sur la base du nombre d'états d'assemblage et du nombre de tâches à exécuter.

La validité et l'applicabilité de la présente méthode proposée est vérifiée en utilisant des exemples pratiques tirés de l'expérience du manufacturier d'appareils électroménagers Frigidaire Canada. La séquence d'assemblage choisie pour assembler une porte de four coïncide exactement avec la séquence présentement en usage chez Frigidaire, laquelle a été développée après plusieurs essais expérimentaux. Ainsi, cet exemple démontre clairement l'efficacité de la méthode pour déterminer la séquence d'assemblage en début de la phase de conception et ce, sans avoir à procéder à des essais.

Sur la base de la stratégie décrite, l'étude présente un système intégré de planification d'assemblage assisté par ordinateur (CIAPS) qui permet la détermination systématique de la meilleure séquence à partir d'une description tridimentionnelle (déterminée par CAO) du produit. En parallèle à cette détermination d'une séquence d'assemblage, le système est aussi capable d'identifier la direction d'assemblage du composant. L'utilisateur peut aussi interagir lors des prises de décisions si des choix difficiles s'imposent. De plus, la possibilité de modifier la conception facilite l'implantation des principes de DFA (conception en vue d'assemblage). Succintement, CIAPS s'avère une approche intéressante qui peut guider le concepteur de produits et l'ingénieur de fabrication dans un contexte où les conceptions de produits et de procédés de fabrication évoluent rapidement. De plus, CIAPS peut parfois être utile lorsqu'il faut implanter les concepts d'ingénierie simultanée.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to Professor Kalyan Ghosh, my thesis supervisor for his friendly guidance, understanding, and constant encouragement during the course of this thesis. It has been indeed a privilege to work under his supervision.

I would also like to extend my indebtedness to Professors Laurent Villeneuve and Roland Maranzana for sparing their time whenever I needed and for many of the friendly discussions with them.

My special thanks are due to Professor Clément Fortin for allowing me to use the computing facilities in his laboratory and also for several fruitful discussions I had with him.

I extend my sincere appreciation to Mr. Yves Saint-Laurent, Vice-president, Frigidaire Canada for accepting this project and allowing me to use their designs in this thesis. I wish to thank Mr. Marcel Mireault and other personnel at Frigidaire who were very helpful to me and extended all their support during my stay at Frigidaire.

It is also appropriate to thank my friends Singu Babu, Anil and Pierre Angers for their help in drawing the figures and translating the abstract into French.

Finally, I wish to acknowledge the Govt. of India, NSERC, Northern Telecom Ltd. and Frigidaire Canada for their financial support during my studies at Ecole Polytechnique.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Assembly is one of the most important activities during the manufacture of a product because of its integrative nature. Important strides have been made in automating the various processes involved in discrete part production by the use of transfer lines, numerically controlled machining cells, integrated metal forming press systems, etc. However assembly is still done manually quite frequently, especially for complex systems such as the automotive engine (19). Hence, it is an area which accounts for a large part of the labor cost. For example, in automotive and in telecommunications industries, 45.6% and 58.9% of the workers are respectively involved in assembly (5), and assembly often accounts for over forty percent of the total manufacturing cost (12). Hence, reducing the manufacturing cost is an important priority at the present time. One way of achieving this is to improve assembly planning, because it is in this early phase of product development that the greatest impact can be made on its manufacturing cost. Because of the current trend of frequent changes of the product design and manufacturing strategies, it is desirable to automate and computerize this planning activity. CAD systems are being increasingly used for product design, and integration of the CAD database with the assembly planning system will be beneficial in improving the overall efficiency.

Computer-aided process planning (CAPP) for assembly involves the preparation of a detailed plan for the assembly of the product using its design as the starting point. The most basic information contained in an assembly plan is the assembly sequence, which lists an order in which the component parts can be assembled to produce the assembly. Assembly sequence planning can be thought of as a generic scientific problem in that the fundamental properties of assembly relations, geometries and operations are used to guide the search for correct, optimal or desirable sequence. The determination of proper assembly sequence to be used is critical, because it affects a number of aspects of the assembly process, such as definition of subassemblies, selection of process equipment, design of special tools and fixtures, etc. The choice of sequence that is adopted plays an important role in determining the characteristics of assembly tasks and the final assembly as well. Since it may be costly to overlook a potential candidate assembly sequence, it is desirable to select a sequence from the set of all feasible sequences.

A typical product can have a very large number (often many thousands) of assembly sequences and this number increases exponentially with the number of components in the assembly. The selection of an assembly sequence from all the available alternatives plays a vital role in determining the characteristics of the assembly task such as the difficulty of the assembly operation, fixturing needs, possibility of part damage during the assembly process, the assembly cost, etc. Therefore, it is very important for the manufacturing engineer to determine a satisfactory assembly sequence from all the feasible ones and within the available resources and facilities.

To determine the feasibility of an assembly sequence, it is necessary to analyze various assembly constraints. These constraints can be broadly classified into two groups: hard constraints and soft constraints (32). Hard constraints are those imposed by the geometry and topology of individual components and the whole assembly. These are product dependent and they do not change unless the design of the product is changed. That is, these constraints can never be violated for any kind of assembly environment and hence the name hard constraints. The other group of constraints are usually product geometry independent and specify additional precedence constraints. They can be different for different assembly environments. The nature of the soft constraints is such that it can be used in decision making by the designer to narrow down the choice to a few good sequences from all the feasible sequences generated based on the hard constraints. Hence they are also called strategic constraints. Because of these geometric and physical constraints, certain components must be assembled before assembling the others. This ordering among the different components of the assembly is called precedence constraint knowledge. Because of the undefined and subjective nature of the generation of assembly sequences and the selection process, few planning aids exist for the analysis of assembly. Until now, the generation of assembly sequences has been done in a largely unstructured manner, and it also demands a lot of expertise and knowledge on the part of the manufacturing engineer. The basic objective of this research is develop an integrated assembly planning system that can select the best assembly sequence directly from the 3-D solid model of the assembly satisfying the precedence constraints and other user specified constraints.

The important research issues in automating the assembly planning include:

- Assembly modeling or assembly representation to describe the geometry and topology of the components, connectivity relations among parts, etc.
- Identification, acquisition and representation of precedence relationships necessary to avoid geometric interference during assembly.

- Generation of all feasible assembly sequences satisfying the precedence relations.
- Representation of the generated assembly plans efficiently for further analysis.
- Determination of optimal assembly sequence minimizing assembly time and assembly cost for a given assembly environment and satisfying some specified strategic factors.

With the objective of developing a generative assembly sequence planning system that can determine the best feasible assembly sequence directly from the 3-D CAD model of an assembly, it is planned in this research to address the above issues in three phases. In the first phase, the solid models of all the components in the assembled state will be created using the PADL-2 solid modeling system. The precedence knowledge will be extracted by employing interference detection feature of PADL-2 to identify the presence of contact and collision-free motion of components. The extracted information will be represented in the form of relational model which provides more detailed information about the assembly.

The second phase deals with the generation of all geometrically feasible sequences considering the hard constraints. The generated assembly plans will then be represented in a convenient way for subsequent analysis.

Finally, in the third phase, first the strategic constraints will be identified. These constraints will be applied to reduce the number of sequences by eliminating the sequences that violate them. The selected sequence will then be subjected to quantitative analysis using the assembly costs estimated by the DFA Toolkit software. Finally, the sequence with minimum assembly cost will be selected as the best sequence that will be used to assemble the product.

Based on the described strategy, an integrated assembly planning system will be designed to perform all these functions.

In order to communicate the stated objectives of this thesis in a thorough and organized manner, it is presented in the following format. Chapter 2 provides the review the various assembly sequence generation techniques and their advantages, limitations and applicability. Three of the most important techniques are discussed in detail. Chapter 3 presents a new three level assembly representation method. The geometric reasoning, assembly precedence knowledge extraction and representation are discussed in Chapter 4. The definition of relational model is modified to include the extracted relational information. Chapter 5 describes the steps in the assembly sequence generation algorithm. Chapter 6 proposes a graphical and a tabular sequence representation scheme to encompass all the generated sequences. The importance of sequence count reduction, criteria to be used for sequence analysis and evaluation of sequences is presented in Chapter 7. Chapter 8 illustrates the various modules of the computer integrated assembly planning system. Chapter 9 shows the application of the current method to practical examples selected from the industry. Finally, the conclusion and scope for future work is presented.

CHAPTER 2
STATE OF THE ART

2.1 INTRODUCTION

A mechanical assembly is typically achieved by a series of operations with a view of creating temporary or permanent attachments between the component parts. The identification and generation of ordered sequence of the operations necessary to produce an assembly is called the assembly planning problem. Due to the inherent complexity in the assembly process planning, it is usually carried out in a hierarchical fashion. For example, in robotic assembly, there are four planning levels in this hierarchy (75): assembly-level planning, task-level planning, manipulator-level planning, and joint-level planning. Extensive work has been done from joint-level planning up to the task-level planning (9,14,17,32,43), but only since the last decade, the literature specifically dealing with the automatic generation of assembly sequences has begun to emerge.

A significant amount of research has been done since the early 1960s on the general process of assembly. Most of the published work on computer-aided assembly and assembly automation is related to the design activity (6,71), line balancing (48) and robot languages for assembly (43,55,72). These languages specifically describe physical constraints of the world model to enable a robot to perform assembly operations precisely. Reported work on assembly planning is not yet very complete and in the last few years, researchers have started looking at the importance of assembly sequence generation methods (1,2,7,10,13,16,24,25,32,34,36,47,50,58,64,66,70), which in turn lead to the development of automatic assembly planning systems (9,14,17,32,41,45,65,67,75).

One of the main problems of the assembly planning is the generation of assembly sequences and the determination of the most promising assembly sequence to achieve cost effective assembly of the product. Because of the geometric and physical constraints among the components of an assembly, certain components must be assembled before assembling others. This ordering among the components is called precedence constraint knowledge of the assembly. The precedence knowledge plays an important role in the generation of assembly sequences and planning of assembly. Part connectivity relationships, part interference during assembly, part tolerances, fixturing requirements, etc. are the factors that constitute the constraints in defining precedence relationships among assembly tasks. The existing techniques of sequence generation differ from each other in the representation of precedence knowledge, acquisition of precedence knowledge, translation of acquired knowledge into assembly sequences, representation of the generated sequences and finally in the selection of the desired sequence. Wolter (76), and Ghosh and Reddy (21) have provided a detailed survey, analysis and comparison of several aspects of these methods.

2.2 CLASSIFICATION

The available literature on assembly sequence planning can be classified into following six major groups based on the representation and acquisition of assembly precedence knowledge and the method of generating sequences.

- Liaison graph methods
- Knowledge-based systems

- Graph theory techniques
- CAD based methods
- Mathematical models
- Other methods

2.2.1 Liaison Graph Methods

The first liaison graph method was developed by Bourjault (7). He proposed a means of obtaining all the precedence knowledge of the assembly from the liaison graph of assembly by answering a series of structured questions. The number of these questions is exponential with respect to the number of liaisons. De Fazio and Whitney (16) simplified Bourjault's method and reduced the count of questions. These methods and their improvements will be explained in detail in the subsequent sections. Both approaches lend themselves to interactive systems in which a computer program generates the questions, a human expert supplies the answers and the program then generates precedence relationships between connections. But, for complex cases, it may become difficult for the human expert to answer these questions and guarantee the correctness of the answers. Errors and oversights in answering these questions lead to erroneous results.

Lee and Shin (41) presented a method for automatic determination of assembly partial orders from a liaison graph representation of assembly through the extraction of preferred subassemblies, with direct connection to assembly cost. The procedure starts with the selection of tentative subassemblies by decomposing a liaison graph into a set of sub-graphs based on feasibility and difficulty of assembly, evaluating each of the tentative subassemblies based on the subassembly selection indices, and then constructing a hierarchical partial order graph by the recursive extraction of subassemblies. Clustering components into subassemblies sacrifices completeness since the sequences that interleave the assembly of parts of different subassemblies cannot be generated.

Mascle and Figour (50) proposed a method to generate some good sequences, but not necessarily all the feasible ones. It starts with the construction of a liaison graph. The contact liaisons between two parts are expressed by a matrix of half degree of liaisons for each mechanical liaison, i.e., two matrices for each liaison. Principal nodes are classified and sorted according to the half degree liaisons. The component with the least half degree liaison value will be disassembled and the matrices are then updated. The process is continued until all the components of the assembly are disassembled, and the reverse of this order is the assembly sequence. A logical module AUTOGAM has been developed to evaluate the effectiveness of the algorithm. Though this method eliminates the combinatorial explosion of sequences, this is not complete as it does not generate all the feasible sequences. There is a possibility of missing a better sequence. Another problem is, with large number of parts (i.e., with larger number of liaisons), the number of half degree liaison matrices become increasingly large which complicates the analysis.

2.2.2 Knowledge-based Systems

Since the task of product assembly involves assembly practice heuristics, knowledge-based systems constitute another approach in assembly sequence planning. In this approach, rules are formulated, based on part stacking properties or part spatial configurations, in order to reason about valid assembly sequences or formation of part groups from the entire assembly. Final sequences are found by recursively applying rules to analyze the parts in the subassemblies or part groups.

Huang and Lee (33) developed a knowledge-based assembly system based on pre-conditions and post-conditions of an assembly state for automatic acquisition of precedence relations. Two algorithms, FIND-PRE-CONDITION and FIND-POST-CONDITION were developed to obtain the knowledge systematically from the feature mating operation graph of the assembly generated from the CAD data of a product. The predicate calculus representation of knowledge about assembly structure, precedence constraints and resource constraints forms the static knowledge base.

Chen and Wichman (14) presented an integrated assembly system which integrates neural network computing and a rule-based planning system (CLIPS). Given a desired assembly in conceptual format, the neural network module of the system retrieves a similar conceptual assembly design from the design memory. Based on the assembly concept, the designer can retrieve a desired 3D B-Rep of the assembly. A feasible assembly plan is then generated minimizing the tool changes for the assembly robot and subassembly reorientation. The system attempts to detect all the component obstructions prior to the disassembly analysis phase. This is an expensive operation. As the individual subassembly operations are considered for disassembly, obstructions of components could be considered only in the disassembly direction under consideration. This will result in better system performance. But, the system considers only two criteria to select the best plan: tool changes and tool directionality, while ignoring many other important criteria.

Sekiguchi et al (66) first analyzed the connective relations such as fit, contact, etc., between the parts of an assembly and the relations are sorted according to the degree of difficulty of disassembly. The sequence is determined primarily by the connecting rules between part groups that are categorized based on the functions of the parts and also rules on the priority of the assembly of the parts within each group. The user must have some knowledge about the design of the product in order to classify the type of connection. Moreover, the degree of difficulty of disassembly is not necessarily the same as that of the assembly.

Delchambre (17) has presented an assembly planner that generates assembly plans by first determining the precedence constraints based on geometric information. The PROLOG implemented system determines the product graph whose links represent the assembly liaisons.

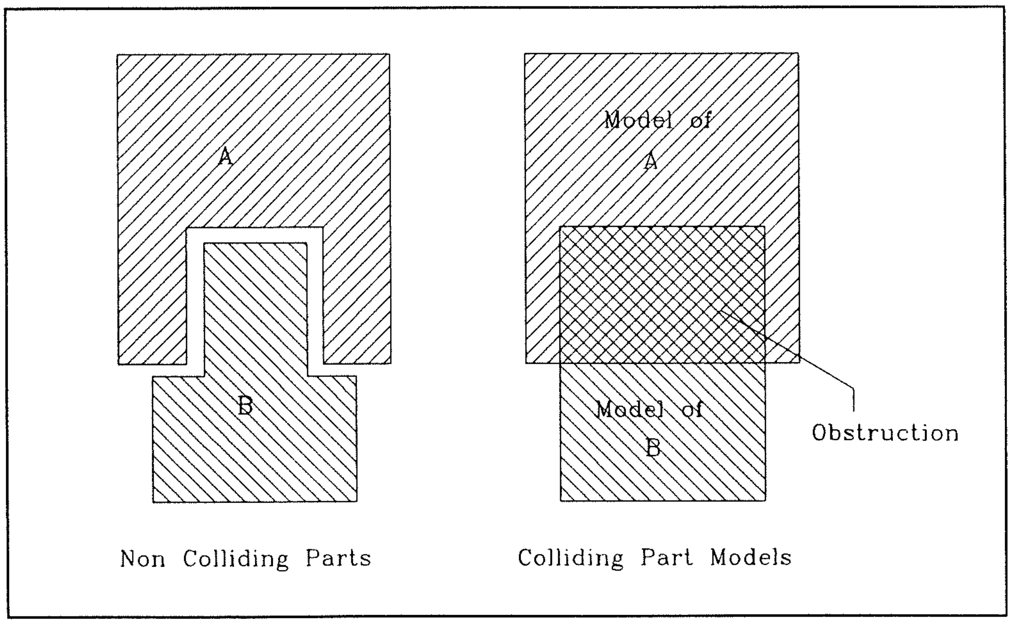


Figure 2.1. Erroneous collision detection (14)

The system represents each part by judiciously disposed parallelepiped whose edges are parallel to the coordinate axes and whose sides enclose the part. The collision detection algorithm of the system uses this simplified part modelling information to determine part obstructions. In some cases this modelling approach can lead to erroneous precedence relationships. For instance, two parts illustrated in Figure 2.1 may freely move toward one another without contact in opposing parallel directions. However, when modelled by parallelepipeds, they obstruct one another. This could result in failure of the system to generate at least one feasible assembly plan.

Chang and Wee (9) used a knowledge-based planning system for mechanical assembly using robots. Using the hierarchical model of the assembly, the determination of assembly sequences is reduced to the problem of ordering the sequence of assembly of various components in each subassembly, and of ordering the sequence of assembly of each subassembly at the same level in the tree. Heuristics are used to achieve this ordering.

The major drawback of these knowledge-based systems is that they are focused on some particular assemblies and assembly cells. Whenever there is a change in the design of the product or in the assembly environment, the knowledge has to be updated.

2.2.3 Graph Theory Techniques

Attempts are also made to determine assembly sequences based on the enumeration of disassembly sequences. One of the approaches is to model the part connections in an assembly graph. In view of the complex interrelationships between the design features of the components of an assembly, assembly graphs provide the designer with a convenient aid for appropriate assembly planning. Graph theory are then applied to cut repeatedly the assembly graph into pieces of sub-graphs. The order of graph dissection is then arranged to represent disassembly sequences. The assembly sequence can also be obtained by removing selected parts or subassemblies one by one from the assembly. Once disassembly sequences are determined, the assembly sequence can be obtained by reversing the assembly sequence.

Homem de Mello and Sanderson (31) suggested a decomposition approach that takes the description of the product and returns an explicit representation of all feasible assembly plans in the form of AND/OR graphs. They decomposed the problem into distinct sub-problems, each one having to disassemble the assembly into two subassemblies or parts. All the decompositions are then combined to obtain disassembly tree. The authors also presented an algorithm (28) which yields the AND/OR graph representation of assembly sequences. However, such trees are relatively large and time consuming to generate. This method will be explained in detail in the coming sections.

Heemskerk and Van Luttervelt (26) also used the disassembly approach introduced by Homem de Mello and Sanderson. The product is modelled in part-relation network, which is equivalent to the Bourjault's graph of connections. However, an important characteristic is that some of the edges in the network represent noncontact relationships between components such as "enclosing" or "blocking". Important features of the work include the use of a clustering algorithm, which iteratively scans the part-relation network and checks for part groups that meet a cluster specification. This pre-processing of the network helps to identify subassemblies a priori. They also proposed the use of heuristics (in the form of accessibility and stability checks) to select good assembly sequences. The generated disassembly sequences are encoded into assembly state transition diagram (ASTD), which is a directed graph of assembly states.

Wolter (75) proposed a constraint graph to implicitly represent the precedence relationships among the movements of the components in an assembly. Feasible assembly sequences can be generated by solving this constraint graph. However, this method cannot handle subassemblies. Sphitalni et al (67) analyzed connectivity graphs to determine possible subassembly candidates for disassembly, with multiple directions allowed.

2.2.4 CAD-based Methods

As geometric modelling systems have replaced manual drafting, some attempts have recently been made to develop CAPP (Computer Aided Process Planning) systems applied to assembly processes. In these systems, the data for the analysis is extracted from the CAD model of the assembly. The precedence relations are obtained in the form of graphs or tables from the analysis of the CAD model. With a similar objective Ko and Lee (37) have developed a method for automatic generation of assembly procedure. Their procedure requires the designer to input the mating conditions (namely against, fits, tight-fits and contacts) which are then represented as a mating graph. Based on these mating conditions, a hierarchy of components is derived which represents a break down of the assembly into subassemblies at various levels. The final assembly sequence is coded in the form of standard precedence graph. Since the designer has to input the mating conditions, this approach is tedious and cumbersome.

Khosla and Mattikali (34) addressed a different methodology to automatically determine the assembly sequence from a 3-D solid modeller description of an assembly based on successive component disassembly. This approach starts with the creation of geometric models of individual components using a solid modeller called NOODLES. Locations and orientations of the components are specified interactively by the user from a set of movement and orientation operators supplied by the modeller. A component graph is created to describe the mating conditions between components. Then, reasoning about geometric model gives a list of disassembly operations which are then recorded in the form of an AND/OR tree, and this tree represents the hierarchy of subassemblies and components. This method relies on repetitive testing of degrees of freedom for each part in the model, which in turn requires investigation of the motions along all potential dimensions. So, this is relatively computationally expensive.

Santochi and Dini (65) described a software system called Flexible Assembly Planning System (FLAPS) for assembly planning starting from the CAD model of the product. FLAPS basically has three modules for sequence generation, assembly operation planning, selection of the sequences and off-line programming of machines. They used a CAD solid modeller (EUCLID) to perform the translations of the components to identify the contacts. All the contacts are tabulated in the form of an incidence matrix. The subgroups are identified by applying a set of rules on the incidence matrix and the contact graph. They attempted to generate assembly sequences for each subgroup and for the whole product by disassembly of the components. Whenever a component is disassembled, the table of contacts must be updated which is a very time consuming process.

Lin and Chang (44) also described a similar approach that transforms a 3-D design of a product into an assembly plan. Part mating and collision information is directly extracted from the design of the product created using a solid modeller (TWIN) and is stored in three graphs: connectivity graph, mating direction graph and spatial constraint graph. These graphs are then converted into an assembly precedence diagram (APD) to generate the assembly sequences. The non-geometric information which is provided through a frame-based symbolic representation is applied to revise the generated plans. Using the same logic they have developed a 3D mechanical assembly planning system called 3D MAPS (45).

Hoffman (27) presented a technique which takes CSG representation of the components along with their relative positions in the assembled state, and the technique identifies a path for separating one object from another. The reverse of this procedure gives an assembly sequence of the product.

Laperriere and ElMaraghy (39) developed a generative assembly planner that can generate robotic assembly sequences from a feature based product database. This database is created interactively on the PADL-2 solid modeller. Assembly relations between components are modelled interactively in the relational diagrams. The initial and final relation diagrams are used to describe respectively the initial and final states of assembly. The validity of the physical connections defined in the final relation diagram is checked by analyzing the information contained in the feature-based product database using validation rules. A single robotic assembly sequence is automatically generated, using the relational data defined in the final relation diagram.

In all these methods, the mating conditions from geometric models are first translated into a graph and then this graph is converted into precedence rules to determine the sequence. This two level translation requires an enormous amount of computation and hence they are computationally quite complex. Analyses of these relational graphs requires the use of complex algorithms.

2.2.5 Mathematical Models

For assembly sequence determination, the travelling salesman algorithm is the only standard mathematical modelling technique that is used to date. Chang and Terwillinger (11) proposed an algorithm for printed wiring board assembly. Its application is limited to electronic assembly, such as IC (Integrated Chip) insertion in printed circuit boards. The IC insertion assumes that all motion during assembly is along the same axis, and the insertion head is moving in a two-dimensional plane from point to point. This method is not flexible enough to solve general assembly sequence problems. Oyama and Abe (54) also described a similar approach using a step-wise clustering method. They treated the assembly sequence problem to be equivalent to the drawing path problem, where the shortest path that passes through points and line segments is to be determined. They considered an assembly operation to be just a pick-and-place operation. In printed circuit board assembly, the robot movements are mainly confined to two dimensional space only, which is not the case with complex mechanical assemblies.

Klein and Rhee (36) has proposed a method that is a combination of heuristics and dynamic programming approach using a recursive formula for assembly sequence determination. A subset generation method is adapted to decompose an assembly problem into several subassemblies. The mating operation time and difficulty function are used to evaluate feasible assembly sequence.

Chen (13) has developed another method coupled with pattern-matching operations to generate all feasible assembly sequences. This method uses concepts derived from the State-Constrained Travelling Salesman Problem to formulate the liaison precedence relationships and to obtain the best assembly sequence. The precedence relations among the components of an assembly are transformed into a pattern matching problem. The concept of the matching algorithm is to match liaisons or parts with one of the answers to obtain the current assembly operation. This algorithm reduces the number of questions asked for the generation of assembly sequences, but it requires a lot of expertise from the user to answer these questions.

2.2.6 Other Methods

There exists many other works on assembly sequence generation, each having its own advantages and limitations. Recently, several experimental assembly planning systems that generate an assembly plan based on some preselected criteria has been developed. Bonschancher and Heemskerk (4) have proposed some heuristics to group the parts into some typical subassemblies. This reduces the complexity of the problem, but some groupings are subject to human interpretation. Most systems utilize a branch-and-bound search technique and graph techniques to implicitly represent the precedence relationships among the movements of the components during assembly.

Zussman et al (77) provided a kinematic approach that can be integrated into assembly planning framework to obtain an optimal assembly sequence. They extended the kinematic pair description of a contact to total degrees of freedom in order to consider the disassembly of the components in all possible directions.

Chang et al (10) attempted to use fuzzy concepts in order to evaluate the feasible assembly sequences that are generated considering contact, geometrical precedence and technological precedence constraints. Each assembly step is associated with a fuzzy weight representing the degree of difficulty of the assembly operation. Aggregation of these fuzzy weights of all the steps of all feasible sequences are then compared to determine the best sequence. The distribution of the fuzzy membership function might be different for different criteria and it becomes increasingly complex with the number of criteria.

In their recent research, Ben-arieh and Kramer (3) have considered the assembly sequence generation as the generation of feasible part arrival sequences. In this method, for each part, all the parts that are in contact must be identified. Then the sequences trees are created considering each part as root node. That means, for an N component assembly N number of trees are to be created. Precedence constraints are then applied to eliminate infeasible sequences. For large number of components, creation of these trees and traversing through these trees to obtain only feasible sequences become cumbersome.

Subramani and Dewhurst (70) proposed a different approach to determine the disassembly sequences of the product by assessing service difficulties. The central idea of this work was to develop a method for the assessment of service and maintenance difficulties in product design. It was developed for a specific purpose and it is not a generalized algorithm.

Pu (57) has explored a progressive approach to solve the assembly sequence problem using case-based reasoning methods. In this method, he retrieves a solution from its case library, which was created to solve a similar problem in the past and then adapts this solution to the new problem. Since the library for the solutions for each specific feature is to be maintained, it needs a lot of memory to store all of them. Moreover, once the database becomes very large, searching for the solution takes a considerable amount of time.

All the preceding methods overlooked a part of the assembly planning problem, because neither group was interested in the exhaustive research of all available data which an experienced sequencer could see. And also, none of these were successful in solving the sequence generation problems for the products containing a large number of parts, say more than twenty, which is very common in industrial products produced at present. There is a need to develop an intelligent assembly sequence planning system that can determine the optimum assembly sequence directly from the CAD model of the product. With a broader goal, this research is aimed at developing a system that can determine optimal assembly plan from the geometric models of an assembly. As the proposed method is rooted in the work of Bourjault (7), De Fazio and Whitney (16), and Homem de Mello and Sanderson (31), these methods will be discussed in detail in the following sections.

2.3 BOURJAULT'S METHOD

Alain Bourjault (7) has presented a method for generating all geometrically possible assembly sequences for any given product. This method begins by creating a network called liaison diagram, utilizing the information contained in the parts list and in the assembly drawing. In the liaison diagram, the nodes represent parts, and the lines between the nodes represent any of certain user-defined relations between parts called "liaisons". A liaison can be a physical part to part contact or connection. Part names are attached to the nodes and each liaison is assigned a unique number, for later use in the generation process. The example used by Bourjault, the assembly of a ball point pen, and the corresponding liaison diagram are shown in Figure 2.2.

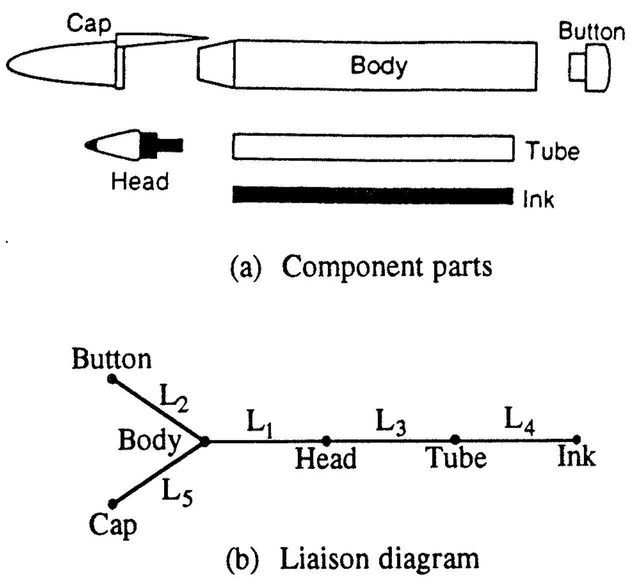


Figure 2.2. An example of a ball-point pen assembly (7)

One can show that the number of liaisons (l) is related to the number of parts (N) by the following inequality,

$$(N-1) \leq l \leq \frac{N^2 - N}{2}$$

In the method described by Bourjault, the component assembly is viewed as the sequential completion of the liaisons between the parts. Thus, the completion of assembly from start can be characterized by a string of numbers representing, in some sequence, the establishment of all the liaisons of the assembly. An important step in the liaison method is the development of rules which describe the possible states of the assembly. The rules and the precedence constraints are the result of a series of questions about each of the liaisons described in the diagram. The response to each of these questions is either "yes" or "no", and this response indicates what subsequent action must be taken. The questions addressed to each of the liaisons are:

Q1: Is it true that the liaison (L_i) cannot be established after the liaisons (L_j, L_k,...) have already been established?

Q2: Is it true that the liaison (L_i) cannot be established if liaisons (L_j, L_k,...) are still not established?

The liaison group (L_i, L_k,...) is called the body of the question. The body can consist of a single liaison or a group of liaisons. The questions of both forms occur in two stages of the technique, called step one and step two; the later step is organized into several modules. Each of the questions of step one addresses only pairs of liaisons. A "no" response to a question in step one results in the omission of the liaison

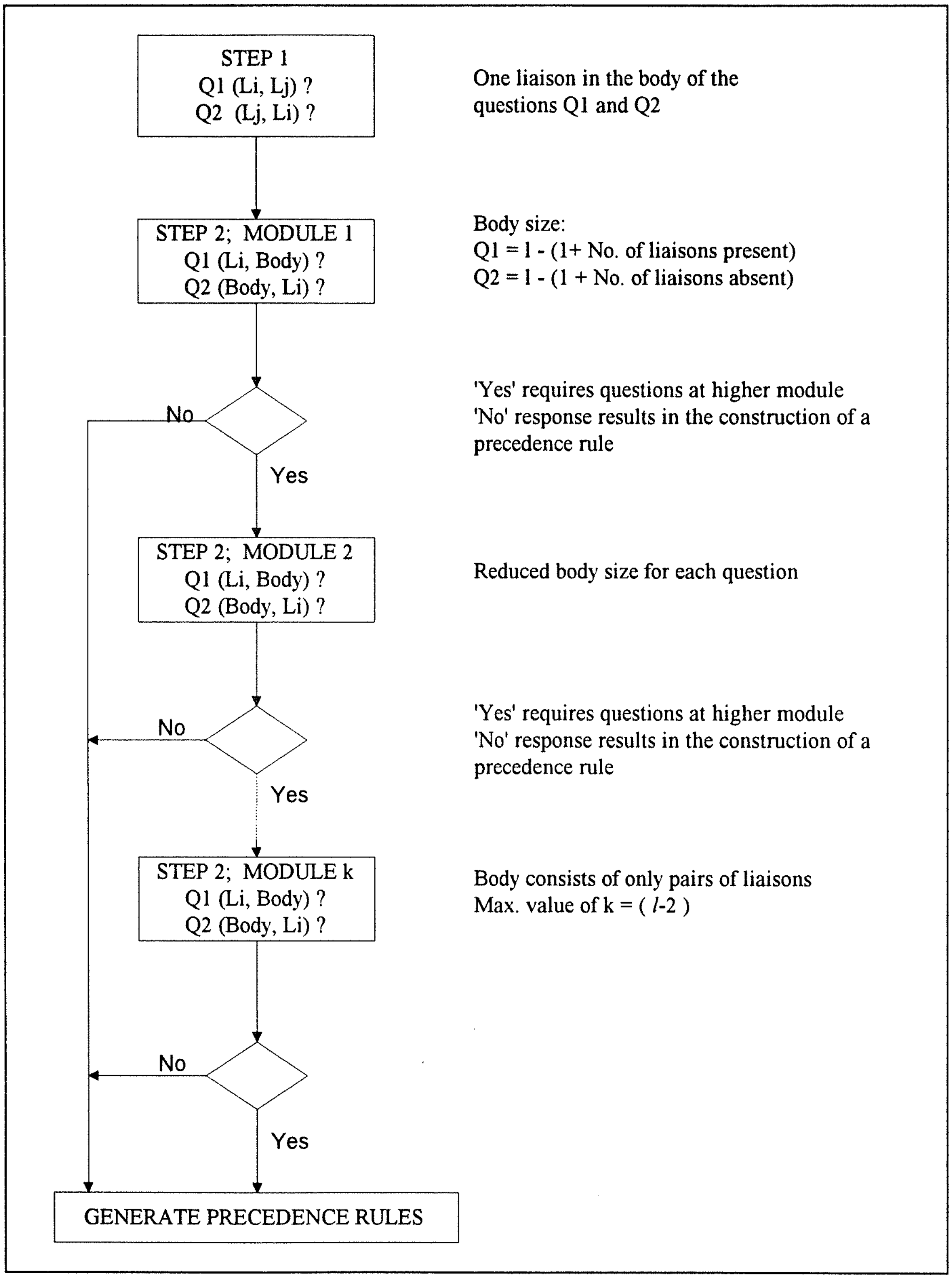

STEP 1
Q1 (Li, Lj) ?
Q2 (Lj, Li) ?
One liaison in the body of the questions Q1 and Q2
STEP 2; MODULE 1
Q1 (Li, Body) ?
Q2 (Body, Li) ?
Body size:
Q1 = l - (1+ No. of liaisons present)
Q2 = l - (1 + No. of liaisons absent)
No
Yes
'Yes' requires questions at higher module
'No' response results in the construction of a precedence rule
STEP 2; MODULE 2
Q1 (Li, Body) ?
Q2 (Body, Li) ?
Reduced body size for each question
No
Yes
'Yes' requires questions at higher module
'No' response results in the construction of a precedence rule
STEP 2; MODULE k
Q1 (Li, Body) ?
Q2 (Body, Li) ?
Body consists of only pairs of liaisons
Max. value of k = (l-2)
No
Yes
GENERATE PRECEDENCE RULES

Figure 2.3. Bourjault's question and answer flowchart (7)

from the body of the question in step two, module one. In that case, the body of the question in step two module one will have (l - 1 - No. of "no" responses) liaisons. The questions of step two module one, if not affected by an answer or answers from step one, have (l - 1) entries in the body, thus there are $2l$ questions in step two module one.

The response to a question in step two dictates that one of two types of action be taken. A "no" response in module one means that a precedence rule or constraint for the assembly may be written. A "yes" response dictates that the questions' progress to the next module, which will have a reduced number of the liaisons in the body. This process continues through higher and higher modules until the questions disappear forming the rules to generate liaison sequences or when the body shrinks to pair of liaisons. The flow chart for the question and answer process is summarized in Figure 2.3. Bourjault logically transforms these rules into statements of which liaisons may be completed under various circumstances. These statements make a complete set in the sense that all valid liaison sequences can be found from these statements.

2.3.1 Sequence Representation

Bourjault represented the assembly sequences in the form of an inverted tree which describes the possible orders of assembly. The origin of the inverted tree is the initial disassembled state (no liaisons established). The next level contains the liaisons which may be completed first. In the case of the ball-point pen they are the liaisons 1, 2 and 3. The next level consists of the liaisons that may follow those identified first. This process is continued until all liaisons have been established. The complete inverted tree representing all possible sequences for the ball point pen is shown in Figure 2.4.

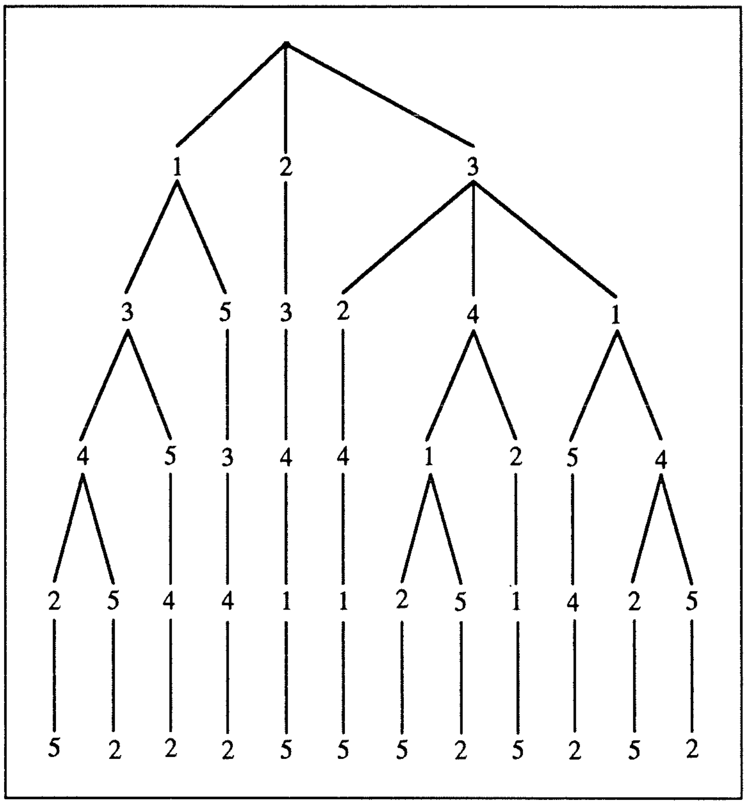


Figure 2.4. Inverted tree representation of liaison sequences for the ball-point pen (7)

2.3.2 Complexity

The question-answer method is designed so that no unnecessary questions are asked. Since no question is redundant and each has a "yes" or "no" answer, the question or answer count is a measure, in bits, of the information content of all the assembly sequences. There are $2l^2$ questions associated with step one ($l \geq 3$ is required for questions to reach beyond step one). Step one defines precedence relations between each pair of liaisons and is the same for any assembly. The number of questions (Q) required in step two varies with the responses at the first level, i.e., all "no" responses in module one complete the definition of the precedence constraints, "yes" responses require additional questions in module two. Therefore, the minimum number of questions associated with module one is $2l$ ($l \geq 4$ is required for questions to reach beyond module one of step two). The limiting case for the number of questions required to define the assembly is dependent on the responses to the questions in module 1 through l, but cannot exceed $l2^l$. Thus the number of questions Q required to properly specify all precedence constraints in the method presented by Bourjault is

$$l2^l > Q \geq 2(l^2 + l) \qquad \text{for } l \geq 3$$

Table 2.1, shown below, exemplifies how quickly the question count grows with the increase in the number of liaisons.

Table 2.1. Minimum and maximum liaison question count (16)

Liaisons	Number of Questions	
	Minimum	Maximum
4	40	64
5	60	160
6	84	384
7	112	896
8	144	2048
9	180	4608
10	220	10240
11	264	22528
12	312	49152
13	364	106496
14	420	229376
15	480	491520

2.3.3 Merits and Demerits

The format and order of the questions guarantee that all interactions and precedence constraints between liaisons are identified. Once all precedence constraints have been identified, enumeration of alternative assembly sequences is also a very straightforward process. The flexibility in the definition of liaison brings a freedom or richness to the technique which an experienced user can exploit to meet specific needs of the assembly under study.

It is the rigour of the question and answer portion of Bourjault's technique that makes its application on assemblies with large part counts cumbersome and tedious. Though, Bourjault's method is algorithmic and forces the practitioner to evaluate all possible relations between pairs and groups of liaisons, it has limited application on more complex assemblies of perhaps seven or more liaisons, because of the number of required questions to be answered.

Henrioud and Bourjault (37) extended Bourjault's theory to deal with geometric and stability constraints. They have developed a software called "LEGA" based on the above algorithm, to evaluate the assembly trees of a given product.

2.4 DE FAZIO AND WHITNEY'S METHOD

Though the sequence generation technique presented by Bourjault is both well structured and rigorous, the sheer volume of questions required to properly define the precedence relations between mates prohibits its application on assemblies with large part counts. De Fazio and Whitney (16) have modified the question and answer portion of Bourjault's method in order to reduce the number of questions. This technique also begins with the construction of a liaison diagram. De Fazio and Whitney altered the format of the questions and they succeeded in reducing their number. They are not "yes-no" questions, and they require geometric reasoning and anticipation by the user. The questions to be answered are:

For i = 1 to *l*

Q1: What liaison(s) must be established to allow establishing liaison L_i?.

Q2: What liaison(s) must be left unestablished to allow establishing liaison L_i?.

These two questions must be answered for each liaison. The answers are directly expressed in the form of precedence constraints between a pair of liaisons or a group of them. The response is represented as a set of Boolean expressions as shown below:

A1: $(L_j \text{ or } (L_k \text{ and } L_m)) \rightarrow L_i$

A2: $L_i \rightarrow (L_s \text{ or } (L_t \text{ and } L_u))$

The symbol "$\rightarrow$" is read simply "must precede". The individual responses can then be combined into one diagram that describes the precedence relationships for the entire assembly. Let us consider the same example considered by Bourjault, i.e., "use and throw" ball-point pen (Figure. 2.2). As in the technique described by Bourjault, the assembly is characterized by a network of nodes and liaisons. The next step in the generation process is the determination of the precedence constraints derived from the responses to the modified questions.

Q1: What liaison(s) must be established to allow establishing liaison L_i?.

Response: $i = 1$, No liaison must be established before liaison L_1 (nothing needs to precede body-to-head mating);

$i = 2$, No liaison must be established before liaison L_2;

$i = 3$, No liaison must be established before liaison L_3;

$i = 4$, $L_3 \rightarrow L_4$. Head to tube must precede ink into tube;

$i = 5$, $L_1 \rightarrow L_5$. Head to body must precede cap to body;

Q2: What liaison(s) must be left unestablished to allow establishing liaison L_i?

Response: $i = 1$, $L_1 \rightarrow L_5$. Identical constraint to that stated above. The body to head mating (L_1) cannot be done if the cap is on the body;

$i = 2$, No liaison must be established, so L_2 may be done;

$i = 3$, $L_3 \rightarrow (L_1 \text{ and } L_2)$. L_3 must be established before both liaisons L_1 and L_2 are established. L_3 does not have to precede them individually;

$i = 4$, $L_4 \rightarrow (L_1 \text{ and } L_2)$. Similar to the one described above;

$i = 5$, No liaison need established so that L_5 can be done;

The complete set of constraints deduced from the above responses can be summarized as shown below:

$L_3 \rightarrow L_4$

$L_4 \rightarrow (L_1 \text{ and } L_2)$

$L_1 \rightarrow L_5$

The next step is to algorithmically generate sequences of the liaisons based on the stated precedence constraints. It begins by determining which liaisons do not have precedents; or in other words, which may be established first. In the case of the ball point pen, by scanning down the right-hand side of the precedence relations, L_1, L_2 and L_3 are identified as first liaisons. Note that the combination of L_1 and L_2 are precedented, but they are individually unprecedented. With liaison L_1 established, L_3 or L_5 may be established next and so forth.

2.4.1 Sequence Representation

While Bourjault uses inverted tree to describe all possible assembly orders, De Fazio and Whitney employed a more compact notation, called Liaison Sequence Graph (LSG), which treats assembly as a series of state transitions starting with a completely disassembled product and concluding with one that is fully assembled. In this diagram, each block represents a single unique state of assembly, with established liaisons being represented by marks in the (numbered) cells. Each line represents a state transition or an assembly move, i.e., the establishment of a liaison. A state transition represents the path from one state to another, and there are usually multiple state transitions leading to and originating from a state. A possible assembly sequence can be visualized as a path from the unassembled state (0^{th} rank) to the final assembled state (last rank).

In Figure 2.5, the first rank shows the completion of liaisons L_1, L_2 and L_3. The next step is to determine the next attainable state of assembly. This is accomplished by evaluating which liaisons may follow each of the first assembly states. For example, once liaison L_1 has been established, the next liaisons which may be completed are liaisons L_3 and L_5. For Liaison L_2, only L_3 can be the next liaison as L_1 can not be done until L_4 has been done. Completing the second rank of diagram, liaison L_3 may be following by either L_1, L_2 or L_4. In this fashion, the user continues marking out the liaison sequences through succeeding ranks through the last rank (fifth in this case). All the twelve possible liaison sequences generated by this process are represented graphically in Figure 2.5.

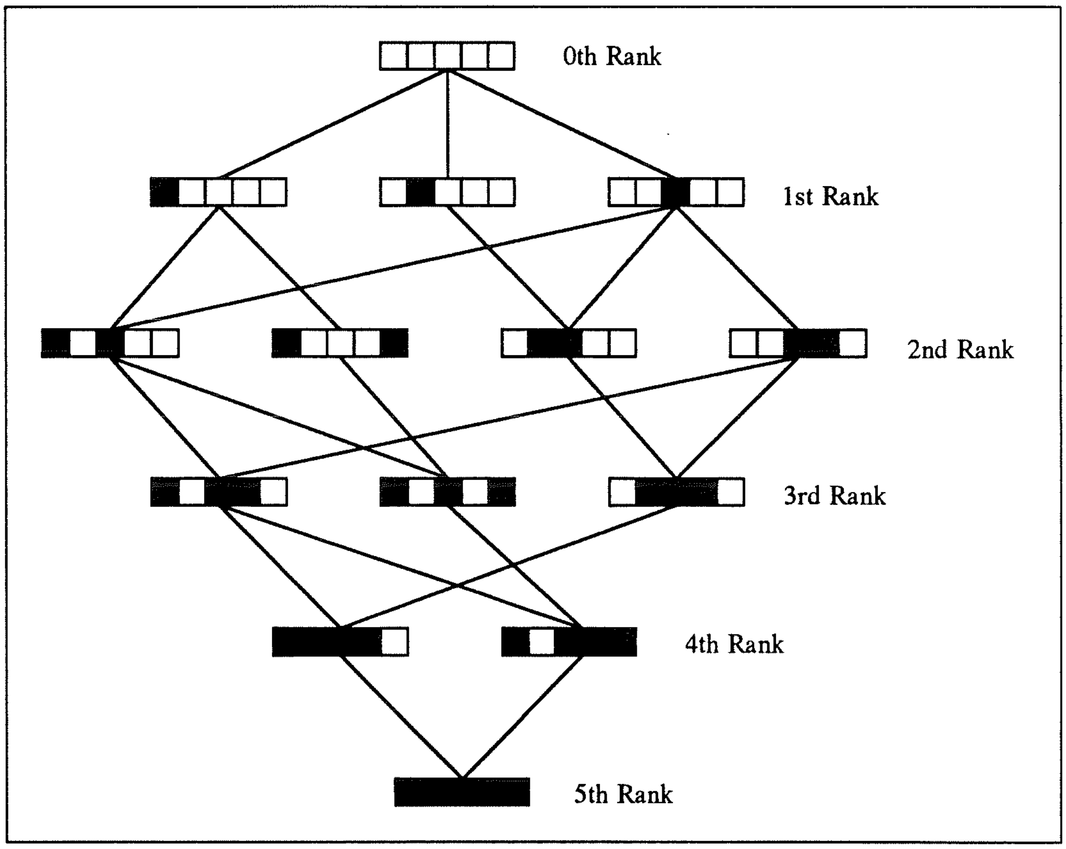


Figure 2.5. LSG representation of assembly sequences for the ball-point pen (16)

2.4.2 Complexity

As only two questions are asked for each liaison, the total number of questions in this method is $2l$. The question count has been reduced from a value between $l2^l$ and

$2(l^2 + l)$. The LSG has the same number of ranks or levels as in the case of inverted tree, i.e., l.

2.4.3 Merits and Demerits

The major advantage of this method is the reduced number of questions required to determine precedence relations. But the response to the modified questions is much more complex, and requires some knowledge of the geometry and anticipation by the user to answer these questions. The state space representation of assembly sequences is better suited to the evaluation of the alternative assembly sequences. Actually, LSG represents the liaison sequences, not the assembly sequences. While dealing with subassemblies, this representation leads to some ambiguity, because establishment of more than one liaison at a time is not always equivalent to assembling a subassembly. Addition of one component might establish more than one liaison.

The other drawback of the liaison sequence analysis is that liaisons are only established for parts which have functional relationships with one another. There are however, many assembly line activities that are not simple part to part mating, typically associated with the establishment of liaisons between parts. Moreover, when liaison count is greater than the part count, there will be loop(s) in the liaison diagram. Therefore, geometric information on the liaison is essential to an efficient sequence generation.

These methods are suitable for manual generation of assembly sequences. An automated version would require the building of a routine capable of identifying the precedence relationships on the basis of topological information. Answering of these questions imposes an extra requirement on the designer, who may view this as redundant, as the information has already been specified in the assembly system.

Klein (35) has made an attempt to modify the liaison representation to incorporate assembly operations, other than part-to-part liaisons, by introducing an additional node called phantom node. Lui (47) has developed a computer program to construct the LSG using the precedence knowledge generated from the answers to the above questions and assembly graph of connections. It also provides a basis for the evaluation and editing procedures. Baldwin et al (1) further extended this method using the cut-set method to represent all geometric and mechanical assembly constraints as precedence relations.

2.5 HOMEM DE MELLO AND SANDERSON'S APPROACH

Homem de Mello and Sanderson (31) developed an algorithm for the generation of assembly sequences which employs a relational model that describes the geometry of the assembly and the attachments that bind one part to another. The relational model includes three types of entities: parts, contacts and attachments, and a set of relationships among these entities. The relational model of an assembly is represented by a graph with its associated attribute functions. In this graph, the vertices are either parts, contacts are attachments and the edges represent various relationships between the pairs of vertices.

The graph of connections introduced by Bourjault is then a simple sub-graph of the relational model of the assembly and it can easily be obtained. Figure 2.6 illustrates the Graph of connections or Relational graph for the pen assembly shown in Figure 2.2. It has six nodes and five connections. Although the ink is not actually rigid, it is assumed as a rigid body.

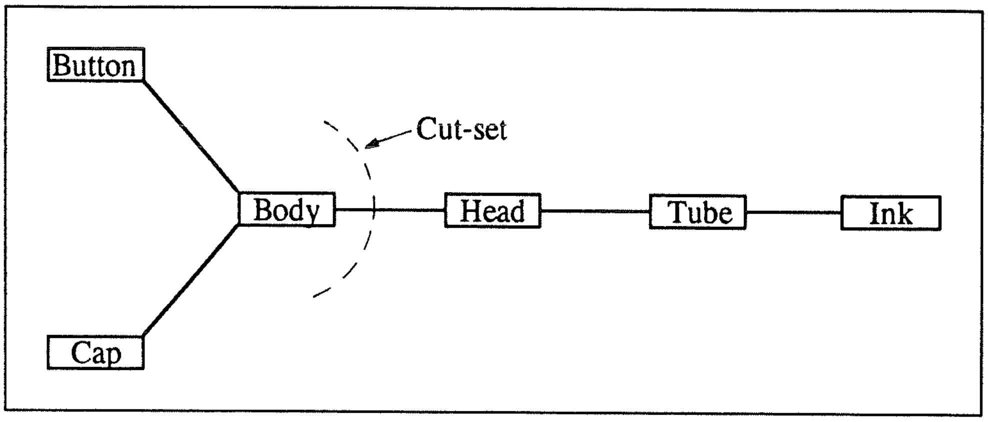


Figure 2.6. Relational graph of the ball-point pen

The algorithm takes a decomposition approach to enumerate the decompositions of the assembly and to select those that are feasible. This decomposition of contacts of an assembly is regarded as a cut-set in that assembly graph of connections. A decomposition of an assembly is feasible only if it satisfies three predicates namely "Geometric feasibility", "Mechanical feasibility" and "Stability". The algorithm is basically divided into three procedures:

1. Procedure "GET-FEASIBLE-DECOMPOSITIONS" takes a description of an assembly and returns all feasible decompositions of that assembly.

2. Procedure "FEASIBILITY-TEST" checks whether a decomposition is feasible or not. A decomposition of an assembly is feasible, if it satisfies the "stability",

"geometric feasibility" and "mechanical feasibility" criteria.

3. Procedure "GENERATE-AND-OR-GRAPH" takes a description of an assembly, and returns the AND/OR graph representation of assembly sequences for that assembly.

2.5.1 Sequence Representation

The knowledge of feasible decompositions allows the construction of the AND/OR graph representation of the sequences. An algorithm for the generation of AND/OR graph of feasible assembly sequences and a proof of its correctness and completeness is presented in (28). Figure 2.7 shows the AND/OR representation of feasible sequences of the ball point pen assembly.

The nodes in this AND/OR graph are the subsets of set of parts P, that characterize stable subassemblies. The hyperarcs correspond to the geometrically and mechanically feasible assembly tasks. Each hyperarc is associated to a feasible decomposition of the subassembly, i.e., it corresponds to a cut-set in the graph of connections or relational graph.

2.5.2 Complexity

The amount of computation involved in this method depends not only on the number of decompositions that must be analyzed which in turn depends not only on the number of parts and how they are interconnected, but on the resulting AND/OR graph as well. The maximum amount of computation occurs for strongly connected assemblies in which every part is connected to every other part (all decompositions of all subassemblies are feasible), while minimum amount of computation occurs for weakly connected assemblies in which there are N-1 connections between the N parts, with the i^{th} connection being between the i^{th} and the $(i+1)^{th}$ parts.

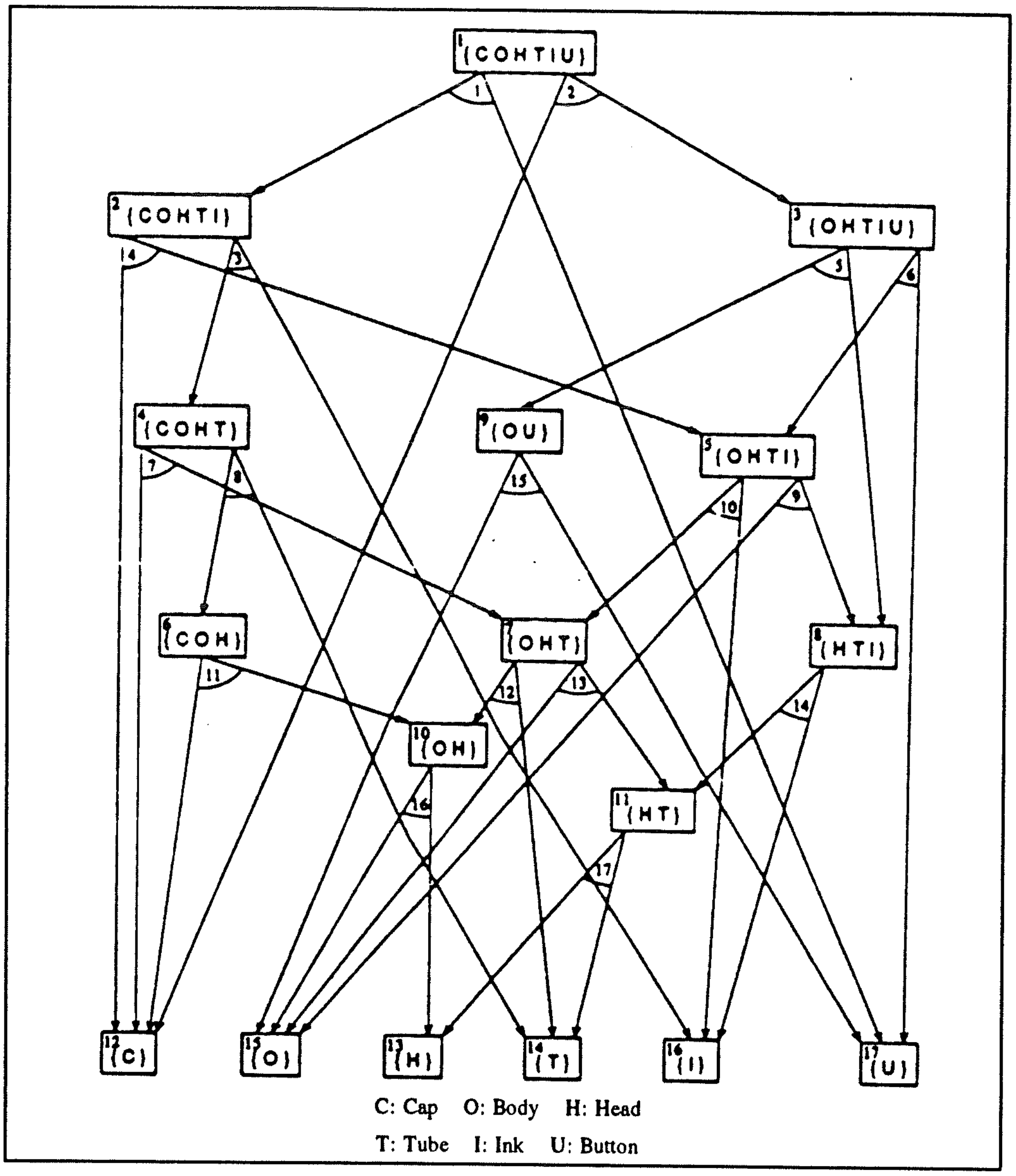


Figure 2.7. AND/OR graph representation of assembly sequences of the ball-point pen (30)

For each possible combination of how the parts are interconnected and for each type of the resulting AND/OR graph, the resulting total number of decompositions D is:

(a). Weakly connected assemblies:

- *One-part-at-a-time tree AND/OR graph* in which at most one hyper-arc leaving each node.

$$D = \frac{N*(N-1)}{2}$$

- *Network AND/OR graph* in which as many hyper-arcs as possible leaving each node.

$$D = \frac{(N+1)*N*(N-1)}{6}$$

(b). Strongly connected assemblies:

- *One-part-at-a-time tree AND/OR graph*

$$D = 2^N - N - 1$$

- *Network AND/OR graph*

$$D = \frac{(3^N + 1)}{2} - 2.$$

The proof for these bounds of computational complexity is given by Homem de Mello and Sanderson (31).

2.5.3 Merits and Demerits

The algorithm is complete and correct. The AND/OR graph uses fewer nodes than the directed graph of assembly states. Furthermore, it explicitly shows the possibility of simultaneous execution of assembly tasks. It does not include any state that cannot be reached from the final assembly (top node), since they do not occur in any feasible assembly sequence.

In this approach, it is assumed that exactly two parts or subassemblies are joined at each time, which is not the case in modular assembly systems. It is also assumed that the sequence of assembly is the reverse of disassembly sequence. Though this is true in most of the cases, it might not be true in some specific instances. For example, the disassembly sequence for repair of a product is not always the reverse of the factory assembly sequence.

Homem de Mello and Sanderson (29) have subsequently introduced two criteria for the selection and evaluation of assembly plans. This algorithm performs a heuristic search for the best assembly plan by maximizing the flexibility of sequencing of the assembly tasks and minimizing the assembly time through parallel execution of assembly tasks. Krishnan and Sanderson (38) have demonstrated a new approach to geometric reasoning about the feasibility of translation motions by introducing an algebra of polyhedral cones which provides a tool for geometric constraints from different part relations.

The approaches presented above, normally require human interaction and judgment to provide crucial sequencing information. Moreover, these approaches used assembly representation schemes which do not provide all the required information for assembly planning and hence they had to use complex reasoning techniques. Because of the complexity of the reasoning involved, it is still impracticable to use existing assembly planning methods to determine the required assembly sequence for assemblies containing large number of parts. A methodology that can eliminate the major drawbacks found in the previous research is still needed. This methodology should use the design of an assembly as input and produce an optimal assembly sequence. Since precedence constraints are determined by the geometry of the product, it is more desirable to obtain the precedence relationships directly from the geometry of the assembly (i.e., from the CAD model of the assembly). The necessity of human interaction in the planning process should be as limited as possible and the generated assembly plans should be useful in real-life applications.

CHAPTER 3

ASSEMBLY MODELING

3.1 INTRODUCTION

A mechanical assembly can be considered as a system consisting of solid bodies constrained by geometric contacts. The geometry and topology of the individual components in the assembly introduce some hard constraints on the configuration of the assembled product and on the order of assembly. It is important to understand the nature of dependencies between parts in an assembly to be able to model the assembly properly. The ability to reason about mechanical assemblies using computer models is limited by the abstraction of the assembly that is created on the computer. The research on high level languages for robotic assembly has explored the use of assembly models. The assembly model must include the spatial positions and the hierarchical relationships among the parts. The trend in the literature for representing the relationships in a mechanical assembly has been to move away from precise geometric descriptions towards more general part/object oriented representation schemes.

3.2 PRIOR ASSEMBLY MODELING METHODS

A number of research papers have addressed the problem of assembly representation. There are three main approaches to the representation of assemblies with three different underlying goals. Lingual representations use a language based representation that is oriented towards representing the objects composing the assembly and the required assembly operations. Graph-based representations, on the other hand, are aimed at further analysis of the assembly to obtain more detailed information. The graph-based representations are usually extracted from a more basic information source such as a CAD database, or from information supplied by the user. Advanced data structures compose the third representation form that utilizes data structures which are focussed on specific issues of the assembly. One of the earliest works on language-based representations for assembly was the RAPT system (55), which avoided the need for a complete volumetric description of the assembly and included only descriptions of the spatial relationships between parts. The relative positions of parts or their degrees of freedom were determined from the set of spatial relationships between them.

Liberman and Wesley (43) developed a system known as AUTOPASS, based on a graph structure in which each node represented a volumetric entity, either a part, a subassembly, or an assembly, and the edges were directed and labeled to indicate four kinds of relationships: "part-of", "attachment", "constraint" and "assembly". The location and orientation of each component is specified by a 4x4 transformation matrix. This representation may require minimum user input. However, it does not provide enough geometric information for the interference checking that is required for assembly procedure generation.

Whereas Liberman and Wesley have only one arc for each pair of parts, Sanderson et al (64) used one arc for each pair of touching surfaces. Each pair of contacts is related by a "contact" relation and each pair of rigidly attached surfaces is related by an "attachment" relation. Srikanth and Turner (68) have used these terms differently. In their work, the word "attachment" unequivocally means that parts are rigidly related, that is, they have no degrees of freedom relative to each other. The words "constraint" or "contact" refer to non rigid relationships.

Most of the approaches represent part relations in the form of a graph. Taylor (72) developed a different scheme for representation of assemblies based on "attribute graphs" that included a volumetric description of parts. Taylor's goal was to find the motions that are needed to bring two parts together, rather than to find the sequences in which parts are put together. Therefore, his attribute graphs described the geometry of the assembly, but did not include any information about attachments that bind the parts together.

Lee and Gossard (40) related the parts by "virtual link". A virtual link is an entity between any pair of mating components in an assembly, and it contains all the information about the mating and the allowed relative motion. In their initial attempt, they have represented only two types of mating: "against" (planar faces butting against each other) and "fits" (requiring axial center lines of individual components to be coincident). Ko and Lee (37) added rigid versions of "against" and "fits" (e.g., "tight-fits" = "fits" + "attachment"). The designer has to provide only mating conditions between mating components to describe the assembly. Rocheleau and Lee (61) suggest some additional mating conditions like "spherical fits" for ball-and-socket joints, "screw-fits" for screw joints, "gear contact" and "rack-and-pinion contact".

Turner (73) gives two relative positioning operators, "face-edge-vertex" (planar contact + attachment) and "coaxial" (coaxial contact + attachment). Unlike Lee and Gossard's "against" and "fit" relations, each of these operators constrains all six degrees of freedom, through alignment of secondary neighboring features.

Dally (15) classified relations into the following types: not attached, jointed attachment (parts move with respect to each other), non rigid attachment (one part depends on the position of another, but not vice versa), and rigid attachment.

Morris and Haynes (51) described twenty different mating feature relationships in terms of the degrees of the freedom constrained by each. While in all the earlier schemes the parts actually touch each other, Morris and Haynes have allowed relations like "face parallel face" and "face parallel edge" which specify that parts can be positioned in space without actually making contact. These constraints are similar to those that would be imposed by assembly tolerances. They define some macro relations like "square peg-in-hole" and "key-ring" that automatically mate several pairs of faces simultaneously and then infer the overall effect on the degrees of freedom.

Nnaji et al (53) in their RALPH (Robot Assembly Language Planner in Harmony) system employed "against", "coplanar", "fits", and "aligned" relations to represent the spatial relationships. Lee and Shin (41) classified the relations as "primary" or "secondary" depending on whether the constrained parts actually touch.

Rossignac (63) proposed a scheme to enhance the semantics of constructive solid geometry (CSG) representations with rigid motions that operate on arbitrary collections of subsolids regardless of their position in the CSG tree. The rigid motions can be used to position the parts in the assembly. Rossignac considers four cases of rigid motion: minimum-distance translation (make two solids come in contact such that their boundaries are tangent at the contact points), minimum-twist rotation, translation about the current axis, and rotation about the current axis. The user picks up boundary features, and the system automatically evaluates the rigid motions. The main disadvantage of this approach is that for any relation between two parts, the user must decompose the relation into a sequence of rigid motions.

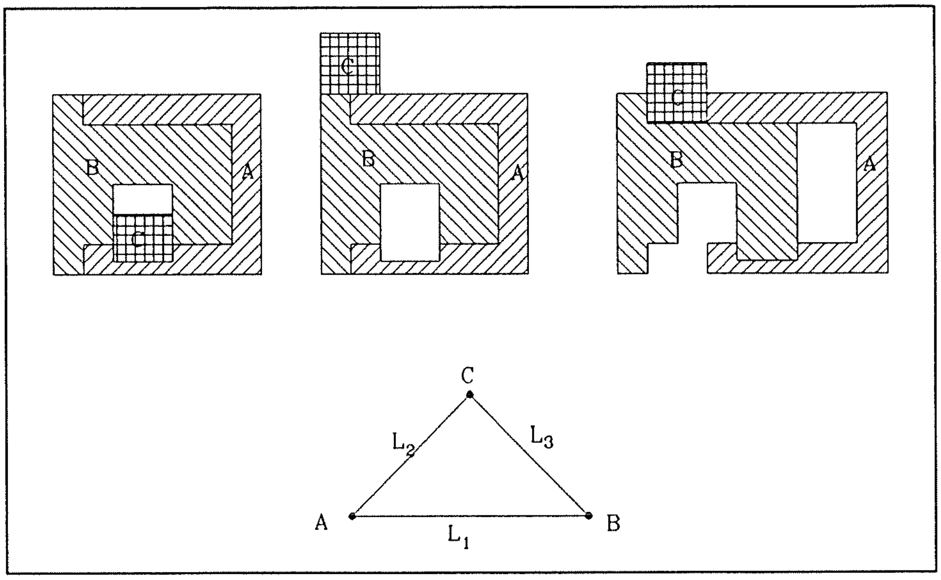


Figure 3.1. Three different assemblies with the same liaison diagram

Unlike the work described above, which aimed at high-level languages for robotic assembly, the work of Bourjault (7) was aimed at modeling the assembly process. As mentioned earlier, he used two types of graphs to represent the products. The "graph of contacts" ("graphique de liaisons mecaniques" (7)) contains one node for each part in the assembly, and one edge for each contact between two parts. Since same pair of parts may have more than one contact, the graph of contacts is not necessarily simple. From the graph of contacts, Bourjault defined the "graph of connections" ("graphique de liaisons fonctionelles" (7)), which has one node for each part in the assembly and one edge for each pair of parts that have at least one contact. By definition, the graph of connections is always a simple graph. The topology of the graph corresponds to the topology of the parts in the assembly. But there is no relation between the geometry of the set of parts in the assembly and the topology of the graph. Figure 3.1 shows three simple assemblies made of the same set of parts. Those assemblies are associated with the same graph, also as shown in Figure 3.1, since the topology of the parts is the same. But the geometric relations in each assembly is very different from the others.

3.3 PROPOSED APPROACH TO ASSEMBLY MODELING

In this research, three levels of description are used for assembly product representation as shown in Figure 3.2. Each one provides more information than the previous one. In this system, the user needs to define only the geometric description of the assembly.

First, as Figure 3.2a illustrates, a complete computer aided description (CAD) of individual part geometries which will be created using a suitable solid modeling system. Second, from the CAD-based description of parts, a relational model of assembly (relational graph) as shown in Figure 3.2b will be derived. This relational model provides an explicit representation of parts and spatial relationships, such as contacts, degrees of freedom of movements among the components of the assembly, etc. Figure 3.2c shows the hierarchical representation of the assembly, based on a particular assembly sequence deduced from the precedence constraint knowledge provided in the relational model.

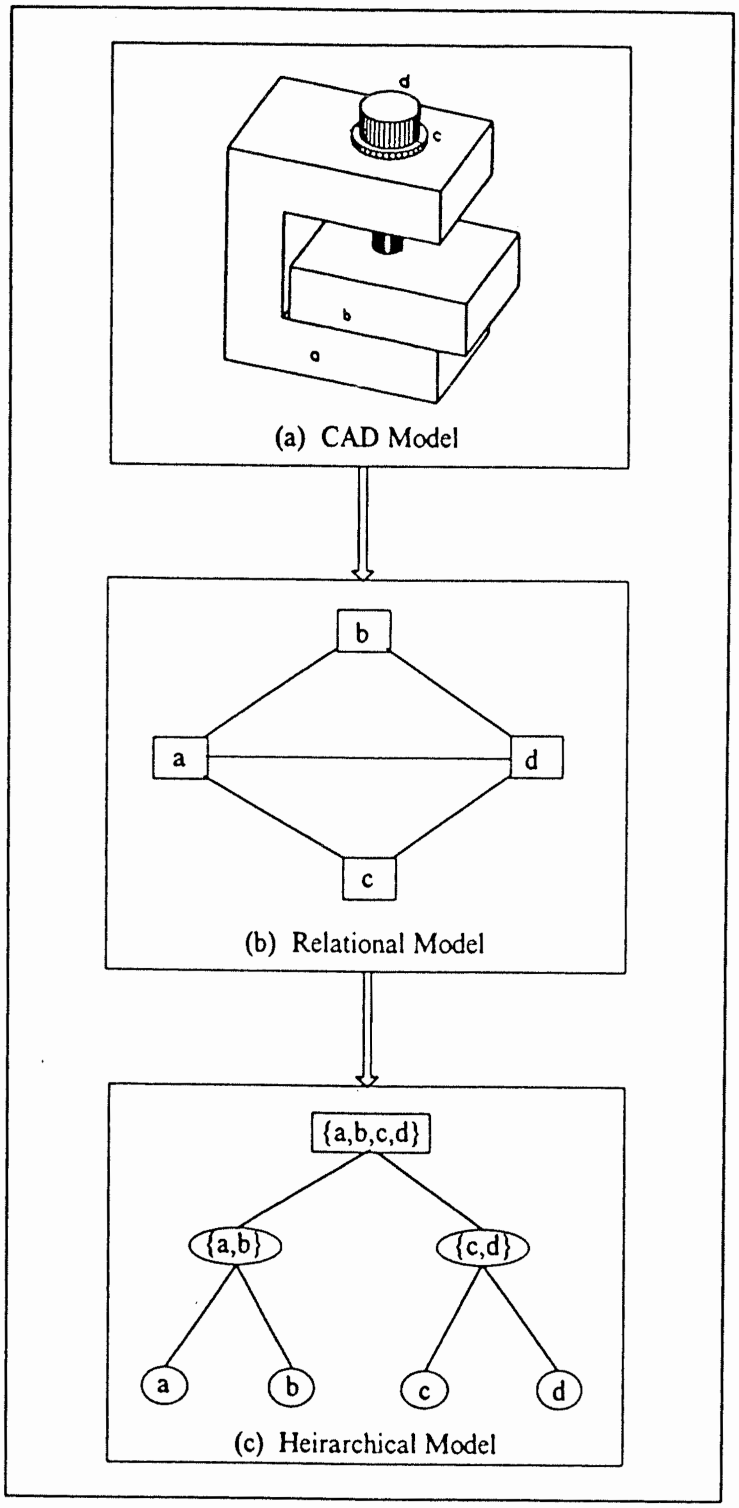


Figure 3.2. Three-level representation of assembly

3.3.1 CAD Model

The starting point of the proposed approach is the CAD model of the assembly, which includes the geometrical and topological data of the assembly objects. Since the main interest of this research is to develop strategies to reason about 3-D objects, solid modeling is chosen to create the geometric models of individual parts of the assembly that represent the geometry and their positions in the world coordinate space. This can be easily accomplished by using a solid modeller such as PADL-2. The topological information that describes relations between the parts of the assembly can be deduced from this CAD description.

3.3.1.1 Solid Modeling in the Proposed System

Solid modeling systems provide facilities for creating, modifying and inspecting models of 3-D solid objects. Mantyala (49) provides a detailed description of the principles of solid modelling and its applications while Requicha and Voelcker (59) describes the important research issues in solid modeling. Representation schemes used by solid modellers fall into one of the following classes: pure primitive instancing, sweeps, cell decomposition (octree representation), constructive solid geometry (CSG) and boundary representation (B-Rep). Each class differs from the other in geometric coverage, informational conciseness, efficiency, validity and usefulness in certain types of applications. Neither representation scheme is uniformly better than the other for all types of applications. For this reason, many systems use multiple representations. However, since PADL-2 uses CSG representation to define the objects, that scheme is chosen in this research.

3.3.1.2 Constructive Solid Geometry (CSG) Representation

In CSG, a solid is represented as a tree of Boolean operators (union, difference and intersection) and its primitives. The leaves or terminal nodes of a CSG tree represent primitive bodies. The branches or non-terminal nodes represent Boolean operations applied to their subnodes, whereas the root represents the object. Figure 3.3 illustrates the CSG representation of a simple assembly model. In this representation, the four cylindrical primitives have parameters defined with respect to the global coordinate system and the final assembly model is the collection of the solid models of the constituent parts. Note that part-1 and part-2 are individual solid models located with designated positions in the assembly. They are then combined to form the assembly using the assembly operator provided in the PADL-2.

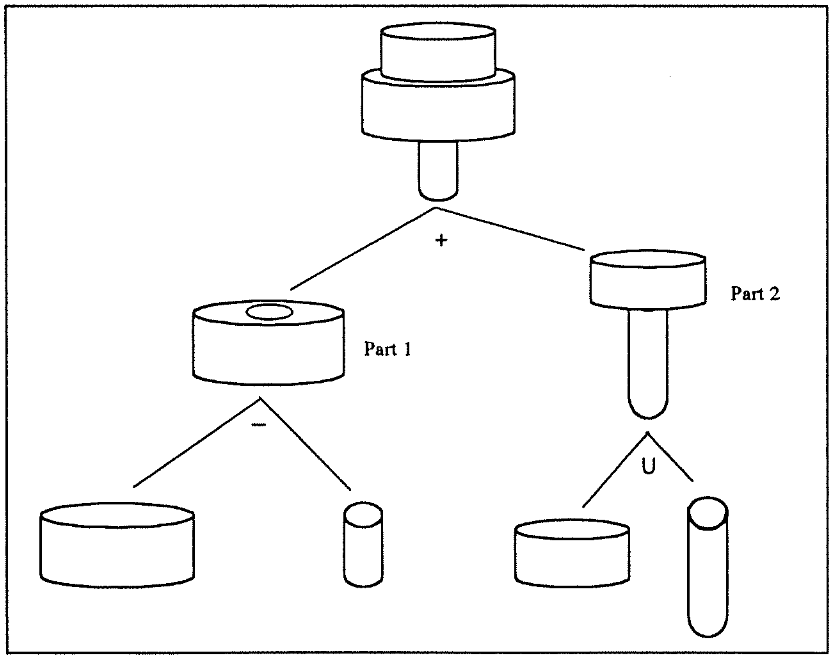


Figure 3.3. CSG representation of an assembly

The major advantage of CSG is that it tells us how to break an object into simpler parts. Furthermore, these representations are fundamental in the sense that they can be converted into other types of representation when necessary. However, the CSG models for a product are not unique as illustrated in the Figure 3.4. But, this has no effect on the representation of the relations among the components of the assembly.

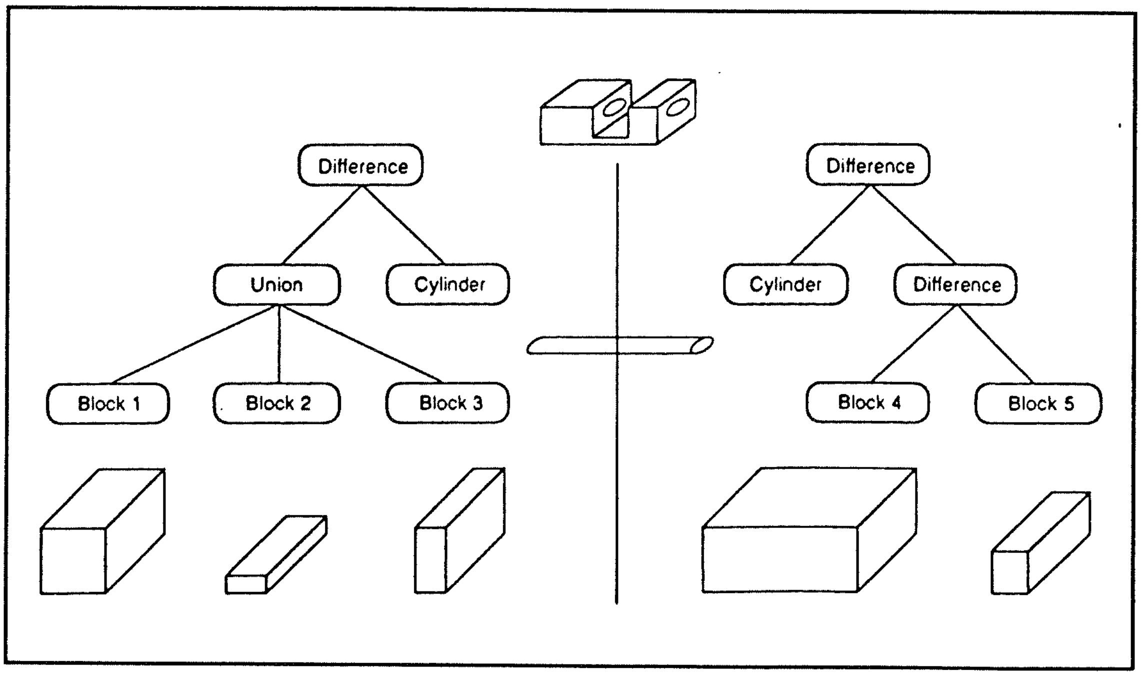


Figure 3.4. Non-unique representation of CSG tree (69)

3.3.1.3 PADL-2 Solid Modeling System

PADL is an acronym for Part and Assembly Description Language. It is a geometric or solid modeling system developed at the University of Rochester. The CSG representation of PADL-2 is used in this work to create the solid models of individual components. In PADL-2, there are five primitives: rectangular block, cylinder, cone, sphere, and torus, which cover 90-95% of industrial products. Liu and Popplestone (46) used Prolog terms to express CSG, which is then translated into the input syntax for PADL-2 by the Prolog predicate "draw". Geleyn (18) used PADL-2 for the simulation of robotic assembly. PADL-2 can also be used to calculate the modeled object properties such as mass, volume, center of gravity and moments of inertia. The characteristics and technical insights into the entities and algorithms that make up the core system of PADL-2 is presented in (14,78).

Many CAD systems also have facilities for dimensioning while a few systems have facilities for tolerancing. We do not currently consider tolerances for planning purposes, although the PADL-2 version available in the Department of Mechanical Engineering at Ecole Polytechnique incorporates them and provides access for use in the planning procedures (60). In addition to the principal components that define the geometry of the product, the attachments such as screws, keys, clamps, etc. can also be represented as separate components.

3.3.2 Relational Model for Assemblies

The CAD model provides enough information for graphic display of the assembly, but it is inadequate for assembly planning. The assembly sequence planning system requires a model of the logical relationships (such as surface contacts and attachments), as well as non-geometric information (such as attachment forces, gravity, friction, etc.) that is not related to part geometry but nevertheless affects assembly methods. An approach proposed by Ghosh (20,22) to represent relations among the components of assembly provide the base in developing this relational model. He had proposed a systematic procedure to represent the geometrical relationships between pairs of components in space, and this approach has been further developed in the present research work.

The relational model can be derived from an existing CAD model. The parts in the model are assumed to be already in place. Relations such as contact and mobility can be derived from the geometry. The relational model itself can serve as an input to a system that can automatically generate assembly sequences. In addition to the geometric information, information about the fastening mechanisms and physical information, such as gravity, stability, etc. may also be included in the model. This makes the relational model more flexible.

The relational model of an N-component assembly is a two tuple $<P,U>$ where,

• $P = \{P_1, P_2, \ldots, P_N\}$ is a set of symbols and each symbol corresponds to one part in the assembly. No two symbols of P correspond to the same component.

• $U = \{U_1, U_2, \ldots, U_M\}$ is set of 6-tuples, representing the relations between components in the assembly, where $M = {}_NP_2 = N(N-1)$.

• $U_i = <P_a, P_b, C_{ab}, T_{ab}, R_{ab}, Y_{ab}>$ where,

$P_a, P_b \in P$,

$C_{ab} = (C_1, C_2, C_3, C_4, C_5, C_6)$ is a 1x6 binary function representing contacts between components a and b.

$C_{ab} : C_i \rightarrow \{0,1\} \quad i = 1 \text{ to } 6$

$T_{ab} = (T_1, T_2, T_3, T_4, T_5, T_6)$ is a 1x6 binary function representing translational motion between components a and b.

$T_{ab} : T_i \rightarrow \{0,1\}$ $\quad i = 1$ to 6

$R_{ab} = (R_1, R_2, R_3, R_4, R_5, R_6)$ is a 1x6 binary function representing rotational motion between components a and b.

$R_{ab} : R_i \rightarrow \{0,1\}$ $\quad i = 1$ to 6

Y_{ab} is the type of contact between the parts a and b, and it takes any element of the set {planar, cylindrical, polygonal, conical, spherical, multiple pegs in multiple holes }, which is a set of most common types of mating tasks in industry. This set can be expanded to accommodate other types of mating tasks.

The C, T and R functions associated with a pair of parts not only provide the assembly precedence knowledge information, but are also useful in determining the type of contact between the pair as well. Though the type of contact (Y_{ab}) is not directly used in the generation of sequences, it will be useful in the design of the assembly process. Detailed description of C, T and R functions will be presented in Chapter 4.

The definition of a relational model representation of assemblies is sufficiently general to encompass a large class of assemblies. The relational model, even with the limitation that it has no hierarchy, can be used for further analysis. Homem de Mello and Sanderson (31) also used the relational model of the assembly as an input to the system that can generate a set of feasible assembly sequences.

The relational model of a subassembly can also be defined in a similar manner by considering itself as an assembly. Since the parts and their relative positions are the same in the whole assembly and in any subassembly, and it was assumed that all contacts between the parts in a subassembly are already established, the subassemblies can be characterized by their sets of parts. Only the relational model of the whole assembly is stored. The relational model of the subassemblies can then be deduced by knowing the set of parts in the subassembly.

3.3.2.1 Attributed Relational Graph

The pertinent information in the relational model of an assembly can be mapped on to a graph defined as an attributed relational graph (ARG) shown in Figure 3.5 which represents the topological structure of assembly system.

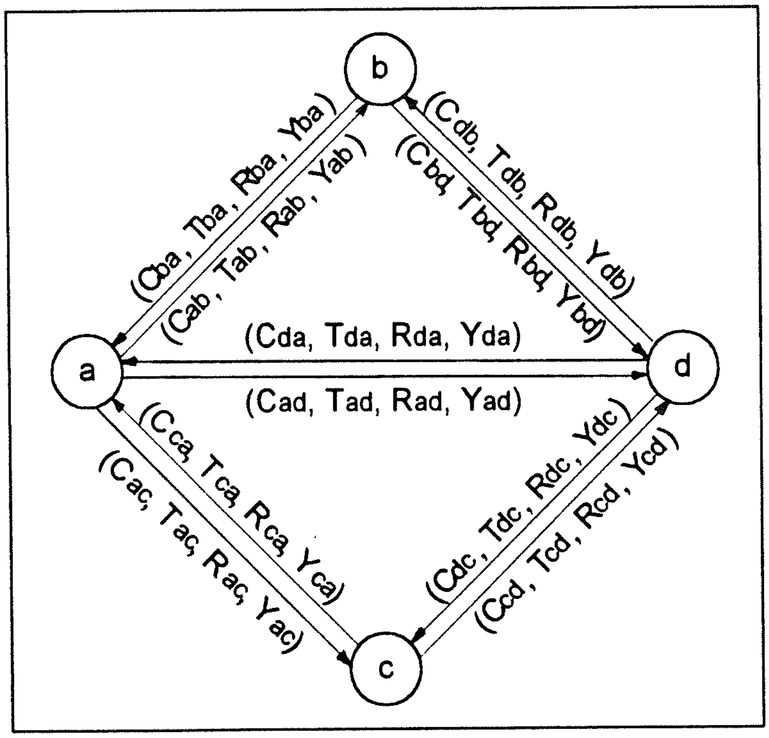


Figure 3.5. Attributed relational graph

The formal form of an ARG can be defined as follows:

$ARG = <P,J,A>$

P = Finite non-empty set of vertices = $\{P_1, P_2,\ldots,P_N\}$

P_i = Identification of part i

N = Number of components in the assembly

J = Set of directed edges = $\{J_1, J_2,\ldots,J_{2m}\}$

J_i = Edge connecting vertices P_v and P_w; $P_v, P_w \in P$

m = Number of mating ordered pairs

A = Set of attributes associated with each directed edge = $\{A_1, A_2,\ldots,A_{2m}\}$

$A_i = \{C_{vw}, T_{vw}, R_{vw}, Y_{vw}\}$, for the edge connecting the vertices P_v and P_w

In this graph, vertices represent the components and edges represent the interactions between the mating parts. The graph is connected, finite and attributed. Each edge is characterized by type of contact, mating directions, and stability, considering all possible local motions (degrees of freedom of motion). The nodes are characterized by the associated attribute function that describes the physical properties of the part. This representation provides a building block for automatic synthesis of assembly operations. Thus, ARG of an assembly contains information on the topology of part configurations, the geometry and relative positions of parts, assembly direction and freedom of motion in part mating.

3.3.3 Hierarchical Model:

The relational model is a one-level description of assembly. The next level of information beyond modeling the relations between parts is to describe the hierarchical structure among the components of assembly. The most natural way of representing an assembly is a hierarchical tree as illustrated in Figure 3.6.

An assembly may be divided into several subassemblies and/or components. Subassemblies are similar to assemblies themselves in that they too can be composed of subassemblies, or basic components. Eventually, all subassemblies can be broken into basic components which cannot be further decomposed. A tree structure is the most appropriate for representing the hierarchical relationships among the various components of an assembly (Figure 3.6). The assembly is the root of the tree, located at the top of the tree at level N, and individual parts are the leaves of the tree at level 1, where N is the total number of levels in the tree structure.

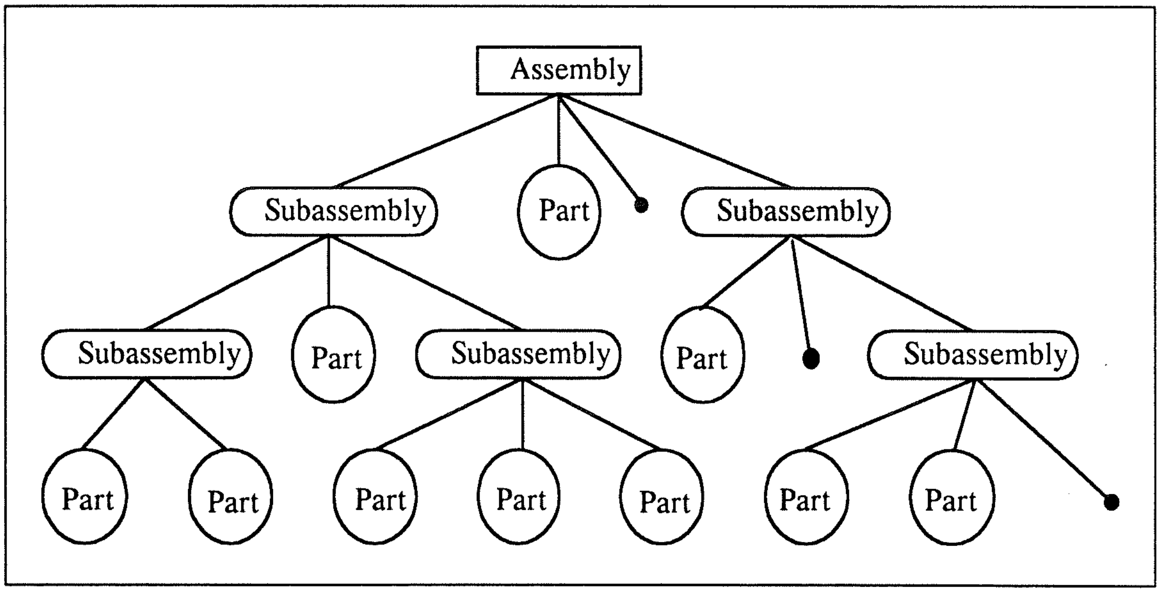


Figure 3.6. Tree structure for a hierarchical assembly model

A hierarchy implies a definite assembly sequence. An assembly may consist of partially ordered subassemblies assembled in parallel. In addition, a hierarchical model can provide an explicit representation of the assembly.

CHAPTER 4
PRECEDENCE KNOWLEDGE EXTRACTION

4.1 INTRODUCTION

The topology, the geometry and the dimensions of an assembly determine the necessary and inviolable constraints on sequence of assembly which can be expressed as precedence relations. The extraction of precedence knowledge is a process of geometric reasoning of the input mechanical assembly models to obtain the necessary information so that the subsequent assembly planning can be done. Thus, the geometric reasoning extracts the precedence restrictions imposed by hard constraints. The information derived from the geometric reasoning includes connectivity relationships, mating directions and collision information. There are two steps to find the relationships among the components of an assembly:

- Determination of the existence of contact between each pair of components.
- Determination of possible disassembly directions of each component from the other.

For any pair of components in an assembly, if there is no contact between them, they are unrelated and are called non-mating parts. For each pair of mating parts, the feasible mating directions to mate these two parts can be found by considering geometric constraints from the mating faces between these two parts. For each pair of non-mating parts, there may exist collision constraints between them. This collision information can be generated by performing collision detection between the solid models of the parts. The collision information for an assembly needs to be determined for both mating and non-mating parts. The reasons are as follows.

- A part in a typical assembly usually has several mating parts. Due to collision constraints, insertion of a part may be prohibited when the assembly procedure between another part and its mating parts is incorrect. For instance, in Figure 4.2, part d mates with three other parts a,b and c. If d is mated with a before b and c are assembled, then b and c can not be assembled, because they collide with a or d. Therefore, it is necessary to determine the collision information between mating parts before correct assembly sequence can be found.

- During the part insertion process, the part being inserted may collide in its travelling path with non mating parts. The discovery of collision information between non-mating parts is therefore needed to determine feasible sequence precedence for parts in an assembly.

In this chapter, first a three-dimensional coordinate system is defined in which, the components of assembly will be created for the purpose of geometric reasoning. Then, how the relations among the components of assembly are represented and how this information is extracted will be discussed.

4.2 DIRECTIONS AND SENSES

In order to reason about sequence planning in an assembly, we need to consider the motion constraints that a particular component can impose by means of its geometric relations to other components. There are two kinds of motion that a component can undergo: translation and rotation. For each type of motion, there are three directions and two senses, hence six motions (six half degrees of freedom) a component can exhibit. The notion of direction and sense must be employed to reason about the spatial relationships that exist among the components of an assembly. The proposed approach operates with components positioned in a tri-orthogonal Cartesian coordinate system as shown in Figure 4.1. Like the direction, sense must also be a discrete quantity in this orthogonal system.

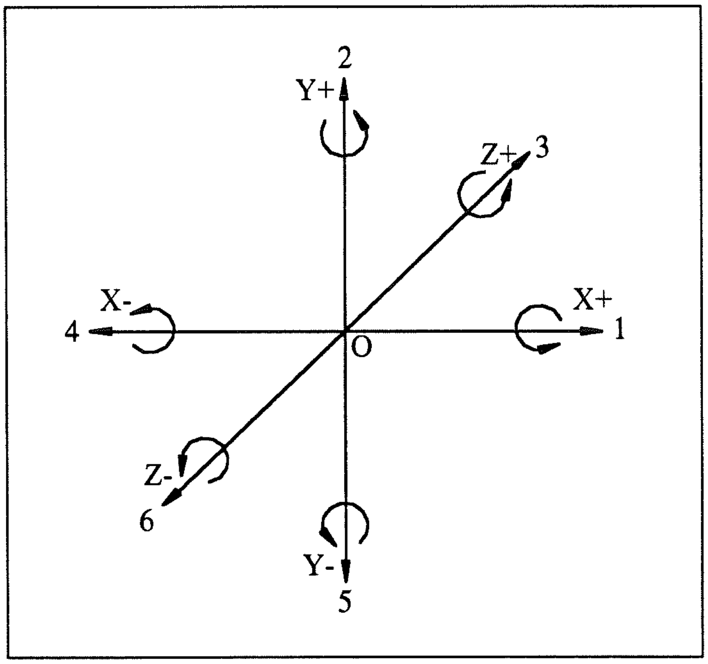
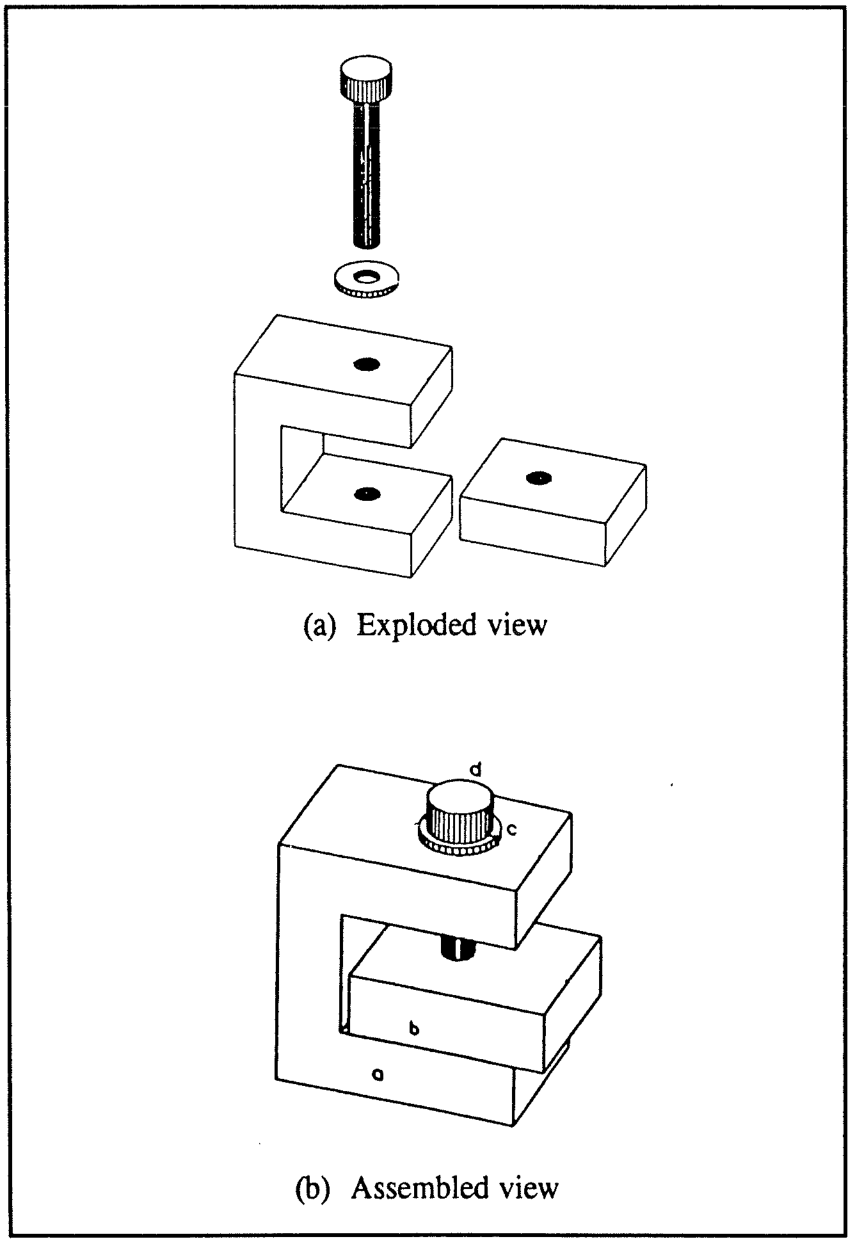


Figure 4.1. Cartesian tri-orthogonal coordinate system

In the Figure 4.1, directions 1, 2 and 3 indicate the positive sense of X, Y, and Z axes (X+, Y+, and Z+) respectively, whereas directions 4, 5 and 6 correspond to the negative sense of X, Y, and Z axes (X-, Y-, and Z-) respectively. The sense of rotation may be found by the right-hand-thumb rule: point the thumb of right hand in the direction along the appropriate axis (e.g., X+ axis) and the fingers curl in the direction of the rotation about that axis (e.g., X+ rotation).



(a) Exploded view

(b) Assembled view

Figure 4.2. Representation of an assembly in 3-D space

Every component must be oriented in either the X, the Y, or the Z direction. Figure 4.2 shows (a) an example assembly, and (b) exploded view of the assembly in 3-D space.

4.3 GEOMETRIC RELATIONS

When two parts are unrelated, they have six degrees of freedom with respect to each other, three rotational and three translational. When a relation is specified between the parts one or more of their degrees of freedom of motion are taken away. For example, when two planar faces are specified to be in contact, one translational and two rotational degrees of freedom are constrained. The parts can still slide and rotate about the plane of contact. When a cylinder is said to fit into a hole, only two degrees of freedom are left (translation along the axis and rotation about the axis). When two surfaces are welded together, these components have no relative degrees of freedom left. Thus, the motion constraints suppress the potential motion of an object. These constraints can be determined on translational and rotational motions of 3-D objects from their contact geometry.

In this work, the qualitative representation of possible connectivities, motions, and how they are constrained in three-dimensional space are represented by three binary functions:

- Contact function (C-function)
- Translational function (T-function)
- Rotational function (R-function).

The C, T and R functions in the relational model, each has six entries in order to consider the three directions of motion and two senses.

4.3.1 Contact Function (C-function)

Two parts are said to be in contact if they are constrained to touch along a surface, line or point. If the components are in surface contact, the components can move relative to one another in the directions along and away from the contact surface. If the components have more than one contact surface, their movement is further constrained. Thus, the possible movements between a pair of components which are in contact can be expressed by a binary function called contact function as follows.

The contact function for a pair of components **p** and **q** in the assembly can be defined as a 1x6 binary function

$$C(p,q) = C_{pq} = (C_1, C_2, C_3, C_4, C_5, C_6),$$

where,

$C_{pq} : C_i \rightarrow \{0,1\}$ $\quad$ $i = 1$ to 6

$C_i =$ 1 indicates presence of contact in the direction i, i.e. part **q** is in contact with part **p** in the direction i

$=$ 0 indicates absence of contact in that direction

C(p,q) can also be written as $C_{pq} = (C_{x+}, C_{y+}, C_{z+}, C_{x-}, C_{y-}, C_{z-})$

4.3.1.1 Extraction of C-function

The values of C-function (1 or 0) can be extracted from the geometry and topology of the parts as explained below.

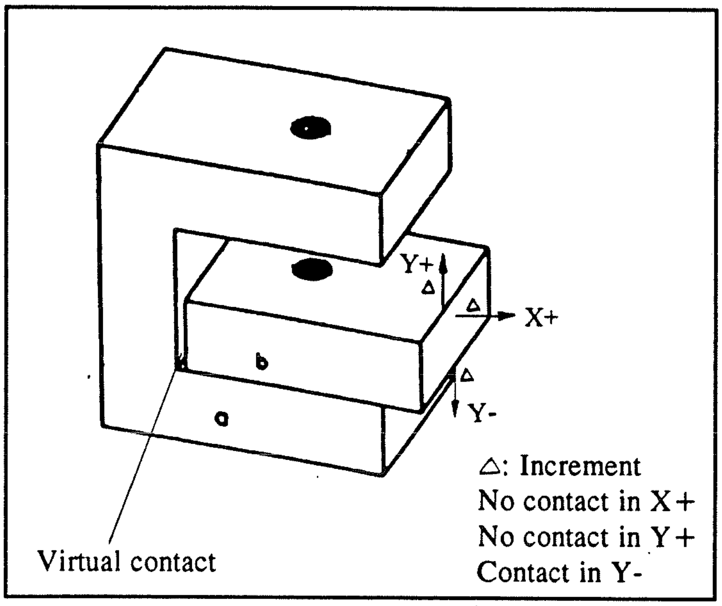


Figure 4.3. Detection of contacts

Consider a planar contact between the parts **a** and **b** as shown in Figure 4.3. Keeping part **a** fixed, if part **b** is moved over a small incremental distance in direction 1 (X+), there will not be any interference between the parts, since **b** has no contact with **a** in direction 1. The first entry of C(a,b) is 0. Similarly, if the part **b** is moved in the direction 4 (X-), there will be some interference between their positions. It shows that part **b** has contact with part **a** in the direction 4. Therefore, the fourth entry of C(a,b) is 1. By performing the same test in the remaining directions, the contact function for this pair can be written as C(a,b) = (0, 0, 0, 1, 1, 0).

In the X- direction, there is some clearance between parts **a** and **b**. Though there is no contact actually, it should be treated as a contact. Hence this is called as virtual contact. The increment value must be chosen carefully, while applying the interference checking on virtual contact. The increment should always be greater than the clearance value.

Applying the same procedure, the contact functions for each pair of components for the example assembly shown in Figure 4.2 are:

C(a,b) = (0, 0, 0, 1, 1, 0) C(b,a) = (1, 1, 0, 0, 0, 0)

C(a,c) = (0, 0, 0, 0, 1, 0) C(c,a) = (0, 1, 0, 0, 0, 0)

C(a,d) = (1, 0, 1, 1, 1, 1) C(d,a) = (1, 1, 1, 1, 0, 1)

C(b,c) = (0, 0, 0, 0, 0, 0) C(c,b) = (0, 0, 0, 0, 0, 0)

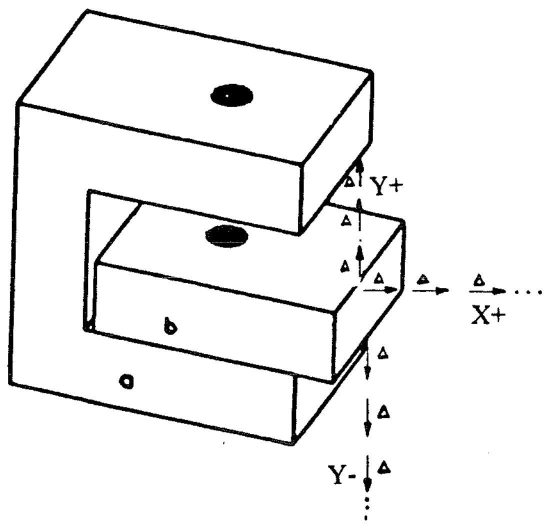
C(b,d) = (1, 0, 1, 1, 0, 1) C(d,b) = (1, 0, 1, 1, 0, 1)

C(c,d) = (1, 0, 1, 1, 1, 1) C(d,c) = (1, 1, 1, 1, 0, 1)

Considering the C-function of parts **a** and **b**, C(a,b) = (0,0,0,1,1,0), which indicates that part **b** has contacts with part **a** in the directions 4 and 5 (X- and Y- directions) respectively. Similarly C(b,c) = (0,0,0,0,0,0); here, all the six entries are zero, which means parts **b** and **c** do not have any contact in any direction, i.e., they are unrelated.

Because of the complex structure of the assembly, the C-function can indicate the presence or absence of contact but does not guarantee the collision free motion of the components. The possible reason is that the parts are moved over a small incremental distance to check for interference. But, there might eventually be interference between the parts, if one of them is moved over a greater distance from the other. Considering the pair of components **a** and **b** only from the Figure 4.2 as shown in Figure 4.4, though there is no contact between **a** and **b** in the Y+ direction, the translation of component **b** is restricted over certain limit. From this, it can be stated that, if there is a contact, obviously there will not be any collision-free path in that direction. On the other hand, in the case of absence of contact, two possibilities arise:

i.e., there may be or may not be a collision-free path in that direction. By comparing the Figures 4.3 and 4.4, in direction Y-, there is a contact between the parts and hence no collision-free path to disassemble the part **b** from **a**. In the X+ direction, there is no contact and there is a collision-free path. In contrast to the preceding cases, in the Y+ direction, though there is no contact between the parts **a** and **b**, no collision-free path exists.



No contact and collision-free path in X+ direction
No contact and no collision-free path in Y+ direction
Contact and no collision-free path in Y- direction

Figure 4.4. Detection of collision-free path

From the above explanation, it can be stated that the contact function provides the necessary condition but not sufficient to assemble two components. To be a feasible assembly operation, it is necessary that there is a collision-free path to assemble the part. That is, the C-function can verify only the local feasibility. To check the global feasibility for assembly, another function called translational function is defined.

4.3.2 Translational Function (T-function)

A component **q** can be dismantled from another component **p** in the a direction i, if and only if there exists a collision-free path in that direction. Let us denote the disassembly or ability of separation of part **q** from another part **p** by a binary function T(p,q) or T_{pq}. T(p,q) can be represented by a 6-tuple composed of the principal axes of motion along which **q** can be separated from **p**. The six entries of T(p,q) represent 6 linear half (3 translational) degrees of freedom along the principal axes of motion.

The translational function for a pair of components **p** and **q** in the assembly can be defined as a 1x6 binary function:

$$T(p,q) = T_{pq} = (T_1, T_2, T_3, T_4, T_5, T_6)$$

where

$$T_{pq} : T_i \rightarrow \{0,1\} \qquad i = 1 \text{ to } 6$$

$T_i = 1$ if the part **q** has the freedom of translational motion with respect to the part **p** in the direction i

$= 0$ if the part **q** has no freedom of translational motion with respect to the part **p** in the direction i

T(p,q) can also be written as $T_{pq} = (T_{x+}, T_{y+}, T_{z+}, T_{x-}, T_{y-}, T_{z-})$

4.3.2.1 Extraction of T-function

The mechanism employed to extract the T-function is exactly the same as the C-function, except in this case, parts are moved over a certain specified distance (usually the maximum distance of a rectangular cube that can encompass the whole assembly). If there is no interference in the disassembly path, the entry of T-function is 1, otherwise it is 0. By applying this procedure to the parts **a** and **b** in Figure 4.3, corresponding T-function is T(a,b) = (1, 0, 1, 0, 0, 1).

For the example assembly shown in Figure 4.2, the translational functions for each pair of the components are:

T(a,b) = (1, 0, 1, 0, 0, 1)	T(b,a) = (0, 0, 1, 1, 0, 1)
T(a,c) = (1, 1, 1, 1, 0, 1)	T(c,a) = (1, 0, 1, 1, 1, 1)
T(a,d) = (0, 1, 0, 0, 0, 0)	T(d,a) = (0, 0, 0, 0, 1, 0)
T(b,c) = (1, 1, 1, 1, 0, 1)	T(c,b) = (1, 0, 1, 1, 1, 1)
T(b,d) = (0, 1, 0, 0, 0, 0)	T(d,b) = (0, 0, 0, 0, 1, 0)
T(c,d) = (0, 1, 0, 0, 0, 0)	T(d,c) = (0, 0, 0, 0, 1, 0)

By comparing the entries of the T-function with that of the C-function, we note that the T-function is simply the complement of the C-function in most of the cases. But this is not always true. For example, if we replace all the 0s by 1s and 1s by 0s of C(a,c), we get T(a,c), whereas this is not true with C(a,b) and T(a,b).

The degrees of freedom of translation (DFT) of part **q** with respect to **p** can be obtained from T(p,q) by DFT(p,q) = $\sum_{i=1}^{6} T_i$. For example, DFT of part **a** with respect to part **b** in Figure 4.2 is $\sum_{i=1}^{6} T_i$ = 3. The DFT signifies the stability of the subassembly. If the DFT is greater, the stability is less. Here "stability" means that the individual components of the subassembly will maintain their relative position during subsequent assembly operations. For example, to compare the stability of subassembly of parts **a** and **b**, and the stability of subassembly of parts **a** and **c**, one can compare

DFT(a,b) and DFT (a,c) whose values are 3 and 5 respectively. As the subassembly of parts **a** and **b** has comparatively smaller DFT value, this subassembly is more stable than that of the parts **a** and **c**.

Translation specifies the position of one part with respect to the other, but not the orientation. In order to exactly represent the position as well as the orientation, the rotational motion of parts (other 3 degrees of freedom) must also be taken into account.

4.3.3 Rotational Function (R-function)

Let us denote the freedom of rotational motion of part **q** with respect to another part **p** by a binary function R(p,q) or R_{pq}. R(p,q) can be written as a 6-tuple composed of the rotation about principal axes of the orthogonal system. The six entries of R(p,q) represent 6 rotational half (3 rotational) degrees of freedom along the principal axes of motion.

The rotational function for a pair of components **p** and **q** in the assembly can be defined as a 1x6 binary function:

$$R(p,q) = R_{pq} = (R_1, R_2, R_3, R_4, R_5, R_6)$$

where,

$R_{pq} : R_i \rightarrow \{0,1\}$ $\quad i = 1$ to 6

$R_i = 1$ if the part **q** has the freedom of rotational motion with respect to the part **p** about the direction i

$= 0$ if the part **q** has no freedom of rotational motion with respect to the part **p** about the direction i

R(p,q) can also be written as $R_{pq} = (R_{x+}, R_{y+}, R_{z+}, R_{x-}, R_{y-}, R_{z-})$.

4.3.3.1 Extraction of R-function

The extraction of the entries of R-function is very similar to that of T-function, except that the motion of the parts is rotational in this case. Another important point to be noted here is that the upper limit of the rotation is known in this case and it is same for any design. Considering the parts **a** and **d** from the Figure 4.2, if the part **d** is rotated with respect to part **a** about Y+ or Y- directions, there will not be any interference. So, the second and fifth entries of R-function will be 1, whereas the rotation in other directions gives interference, the corresponding entries will be 0. The corresponding R-function will be R(a,d) = (0, 1, 0, 0, 1,0).

For the example assembly shown in Figure 4.2, the rotational functions for each pair of the components are:

R(a,b) = (0, 0, 0, 0, 0, 0)	R(b,a) = (0, 0, 0, 0, 0, 0)
R(a,c) = (0, 1, 0, 0, 1, 0)	R(c,a) = (0, 1, 0, 0, 1, 0)
R(a,d) = (0, 1, 0, 0, 1, 0)	R(d,a) = (0, 1, 0, 0, 1, 0)
R(b,c) = (1, 1, 1, 1, 1, 1)	R(c,b) = (1, 1, 1, 1, 1, 1)
R(b,d) = (0, 1, 0, 0, 1, 0)	R(d,b) = (0, 1, 0, 0, 1, 0)
R(c,d) = (0, 1, 0, 0, 1, 0)	R(d,c) = (0, 1, 0, 0, 1, 0)

In this research, R function is not used for the generation of assembly sequences, but it is useful for other mechanical analysis of assemblies such as kinematics analysis. The degrees of freedom of rotation (DFR) of part **q** with respect to **p** is obtained from R(p,q) by DFR(p,q) = $\sum_{i=1}^{6} R_i$. For example, DFR of part **a** with respect to part **d** in Figure 4.2 is $\sum_{i=1}^{6} R_i = 2$. The DFR also has some significance concerning the stability of the subassembly.

Now, the total degrees of freedom (TDOF) of motion is the sum of DFT and DFR. In fact, a subassembly with higher value of TDOF is less stable, but it more flexible as the assembly is possible in more number of directions. For the parts which are rigidly attached (ex. welded parts), the TDOF is zero, i.e., they act as a single component.

The C, T and R functions together give detailed information about the type of contact (such as planar, cylindrical, etc.) which is required for assembly process design. For example, Morris and Hynes (51) described a robot assembly system in which the assembly relations are characterized on the basis of the relative degrees of freedom of motion that are constrained. Thus, the C, T and R functions together provide a unified representation of mechanical assemblies.

Another important aspect that can be observed here is, in any of these three functions, the inverse of the function can be obtained by simply interchanging the left half of the function with the right half. For example, if C(a,b) = (0,0,0,1,1,0) is known, the entries of C(b,a) can be obtained by interchanging (1,1,0) and (0,0,0), i.e., C(b,a) = (1,1,0,0,0,0). This avoids the duplication in performing the collision detection and saves computer time considerably, as it does not require to perform collision detection algorithm to get the values of C(b,a).

CHAPTER 5
GENERATION OF ASSEMBLY SEQUENCES

5.1 INTRODUCTION

The extraction of the precedence knowledge discussed in the previous chapter is a creative step in the generation of assembly sequences. This chapter describes a systematic algorithm to translate precedence constraint knowledge imposed by the geometry and topology of the components of an assembly into assembly sequences. In this algorithm, attempts are made to build the subassemblies starting with two components, and subsequently a component or a previously formed subassembly is added until the assembly of N components, i.e., the final assembly is achieved. The algorithm basically has two procedures. In the first procedure, all the feasible assembly pairs are formed. The second procedure generates all the feasible higher order subassemblies. In each procedure the following constraints will be considered.

- Connectivity constraints
- Precedence constraints

The connectivity constraints specify which parts are connected to other parts in terms of an assembly operation. For example, if part c is connected only to part d, assembling the parts in the order {a, c, b, d} will yield an infeasible assembly sequence. In this case, part c is introduced after part a and it cannot be assembled because it is not in contact with part a. The precedence constraints represent the fact that some components have to be assembled before the others; otherwise, they will interfere with a later assembly operation.

Assuming only one component or subassembly is added at a time, an assembly of N components can be assembled from any one of its additive couples[1](3) as shown below.

((N-1, 1), (N-2, 2),...,(N/2, N/2) if N is even

((N-1, 1), (N-2, 2),...,((N+1)/2, (N-1)/2) if N is odd.

For example, a five component assembly can be assembled in two combinations, (4,1) or (3,2). The 4 component subassembly can be built in two ways: (3,1) or (2,2).

5.2 ASSEMBLY SEQUENCE GENERATION ALGORITHM

To explain the steps of the proposed methodology, a table of pairs, the C-functions and T-functions of the assembly shown in Figure 4.2 are tabulated as shown in the Table 5.1. The entries of these C and T functions from this table for the required pairs will be represented in the form of truth tables, and then methods of Boolean Algebra (62) will be applied to verify the feasibility of the assembly.

The algorithm is divided into two procedures. Procedure I starts with forming all feasible pairs or two component assemblies by checking the contact functions only. In Procedure II, it checks for the feasibility of adding a component or a subassembly to the existing subassembly and it continues this process until the final assembly is formed. Consideration of the stability constraints will be explained in Chapter 7.

[1]*Additive couples of a number N are two numbers whose sum equals N*

Table 5.1. C & T functions for the assembly shown in Figure 4.2

Pair	C-function						T-function					
	C_1	C_2	C_3	C_4	C_5	C_6	T_1	T_2	T_3	T_4	T_5	T_6
(a,b)	0	0	0	1	1	0	1	0	1	0	0	1
(a,c)	0	0	0	0	1	0	1	1	1	1	0	1
(a,d)	1	0	1	1	1	1	0	1	0	0	0	0
(b,c)	0	0	0	0	0	0	1	1	1	1	0	1
(b,d)	1	0	1	1	0	1	0	1	0	0	0	0
(c,d)	1	0	1	1	1	1	0	1	0	0	0	0
(b,a)	1	1	0	0	0	0	0	0	1	1	0	1
(c,a)	0	1	0	0	0	0	1	0	1	1	1	1
(d,a)	1	1	1	1	0	1	0	0	0	0	1	0
(c,b)	0	0	0	0	0	0	1	0	1	1	1	1
(d,b)	1	0	1	1	0	1	0	0	0	0	1	0
(d,c)	1	1	1	1	0	1	0	0	0	0	1	0

5.2.1 Generation of Two-component Subassemblies (Procedure I)

For each pair of components, at least one entry of the contact function of that pair must be 1 to make that pair a feasible subassembly. In other words, the boolean sum of contact function: $(C_1 \vee C_2 \vee C_3 \vee C_4 \vee C_5 \vee C_6)$ should be true.

For example, from Table 4.1, the contact function for the pair (a,b) is C(a,b) = (0, 0, 0, 1, 1, 0) and the boolean sum = $0 \vee 0 \vee 0 \vee 1 \vee 1 \vee 0 = 1$ (true). Therefore, (a,b) is a feasible subassembly. Applying the same principle for all the pairs in the Table 4.1, the feasible two component subassemblies are:

(a,b) (a,c) (a,d) (b,d) (c,d) (b,a) (c,a) (d,a) (d,b) (d,c)

Note that, as only two parts are considered, there is no necessity to check for precedence constraints.

5.2.2 Generation of Higher Order Subassemblies (Procedure II)

In this procedure, we attempt to add another component or subassembly to the subassemblies formed in the previous steps. This procedure has two steps.

Step 1: To add a new component or subassembly, it should have a contact at least with one of the components of the subassembly that has already been assembled. To explain this, consider a feasible subassembly from the previous step, say (a,b). Let us attempt to add the component c to this subassembly. The feasibility is verified as follows.

Consider the subassembly (a,b) as a set {a,b} and the component to be added as another set {c}. The Cartesian product of these two sets ({a,b} x {c}) gives a set of ordered pairs {(a,c), (b,c)} involved in that subassembly. If the contact functions of these ordered pairs have at least one entry of 1, then part c has contact with subassembly (a,b). The contact functions of pairs (a,c) and (b,c) are :

$$C(a,c) = (0, 0, 0, 0, 1, 0) \quad \text{and}$$
$$C(b,c) = (0, 0, 0, 0, 0, 0).$$

Since there is an entry of 1 in C(a,c), the component c has contact with the subassembly (a,b). Mathematically, these values of the C-functions of the corresponding pairs can be written in the form of truth tables and then the "or" operator (operator "$\vee$") can be applied to get the resultant truth values of C-functions (TC) as explained below.

Pair	C_1	C_2	C_3	C_4	C_5	C_6
(a,c)	0	0	0	0	1	0
(b,c)	0	0	0	0	0	0
TC	0	0	0	0	1	0

Here, $TC_i = C_i(a,c) \vee C_i(b,c)$ for $i = 1$ to 6

$TC_1 = 0 \vee 0 = 0$

.

.

.

$TC_5 = 1 \vee 0 = 1$

$TC_6 = 0 \vee 0 = 0$

Now, again applying the "or" operator over the TC, the resultant boolean sum of TC is

$$RTC = TC_1 \vee TC_2 \vee TC_3 \vee TC_4 \vee TC_5 \vee TC_6$$

If RTC = 1 (true), then there is a contact between the parts. For the example discussed above, $RTC = 0 \vee 0 \vee 0 \vee 0 \vee 1 \vee 0 = 1$ (true). Therefore, there is a contact between c and (a,b).

The presence of contact is the necessary condition, but it does not guarantee the non-feasibility of assembly imposed by the precedence constraints. For example, consider the assembly of the component b to the subassembly (a,d). The Cartesian ordered pairs of the two sets {a,d} and {b} are (a,b) and (d,b). The corresponding contact functions are C(a,b) = (0, 0, 0, 0, 1, 0) and C(d,b) = (1, 0, 1, 1, 0, 1). The entries of these two functions show that part b has contact with the subassembly (a,d). But part b can not be assembled after assembling part d to a. That is, step 1, performs the local feasibility test only.

Step 2: This step performs a global feasibility test for the assembly. To consider the precedence constraints, the T-functions of the Cartesian ordered pairs are represented in the form of truth tables. Then the "and" operator is applied to each column of the truth table to find the boolean product of truth values (TT). The entries of TT show the disassembly directions. Considering the assembly of part c to the subassembly (a,b), the corresponding truth table and TT are shown below.

Pair	T_1	T_2	T_3	T_4	T_5	T_6
(a,c)	1	1	1	1	0	1
(b,c)	1	1	1	1	0	1
TT	1	1	1	1	0	1

In the above truth table, $TT_i = T_i(a,c) \wedge T_i(b,c) \qquad i = 1 \text{ to } 6$

$$TT_1 = 1 \wedge 1 = 1$$

$$\vdots$$

$$TT_5 = 0 \wedge 0 = 0$$

$$TT_6 = 1 \wedge 1 = 1$$

The component c can be assemblable, if at least one entry of TT is 1. In other words, the boolean sum of the resultant TT, i.e. RTT = $TT_1 \vee TT_2 \vee TT_3 \vee TT_4 \vee TT_5 \vee TT_6$ should be 1. In the above truth table, since the TT has 5 entries of 1, the component c has five collision-free disassembly directions with respect to the subassembly (a,b). That is, the subassembly (a,b,c) is feasible and the component c can be assembled in any of the complementary directions of these five directions. Similarly, the other feasible three component subassemblies of the example assembly are:

(a,b,c) (a,b,d) (a,c,b) (a,c,d) (c,d,a) (c,d,b)

(b,a,c) (b,a,d) (c,a,b) (c,a,d) (d,c,a) (d,c,b)

It can be observed that, at least one entry of TT will be 0. If all the six entries of TT are 1, that means parts do not have any relationship, i.e. there is no contact between them. If there is no contact, it will be eliminated in the first level.

It can also be observed that the higher order subassemblies attainable from the subassemblies (a,d), (b,d), (d,a), and (d,b) are not included in the above list as they do not satisfy the precedence constraints. For example, the truth table for the subassemblies (a,d) and (b) is

Pair	T_1	T_2	T_3	T_4	T_5	T_6
(a,b)	1	0	1	0	0	1
(d,b)	0	0	0	0	1	0
TT	0	0	0	0	0	0

In the above truth table, all the entries of TT are zero which shows that b does not have any collision free assembly direction with respect to the subassembly (a,d). So, the subassembly (a,d,b) is not feasible.

Applying the same logic to the subassembly (a,b,c) and the component d, the truth tables of corresponding C-functions and T-functions are:

Pair	C_1	C_2	C_3	C_4	C_5	C_6
(a,d)	1	0	1	1	1	1
(b,d)	1	0	1	1	0	1
(c,d)	1	0	1	1	1	1
TC	1	0	1	1	1	1

Pair	T_1	T_2	T_3	T_4	T_5	T_6
(a,d)	0	1	0	0	0	0
(b,d)	0	1	0	0	0	0
(c,d)	0	1	0	0	0	0
TT	0	1	0	0	0	0

From the above truth tables, it can be said that the assembly of (a,b,c) and (d) is feasible. So, one feasible assembly sequence is (a,b,c,d). Similarly all feasible linear sequences are:

(a,b,c,d) (a,c,b,d) (b,a,c,d) (c,a,b,d)

5.2.3 Generation of Sequences with Subassemblies

During the assembly of a product, frequently subassemblies are first produced which are then assembled together. This approach is preferred due to various considerations, such as the desirability of having common modules for a family of products and reducing the length of an assembly line in order to increase its operational reliability.

The logic for the generation of the sequences with subassemblies is the same as for the linear sequences. Instead of adding one component at a time, some subassemblies can also be added. For example, consider the assembly two subassemblies (a,b) and (c,d). The Cartesian product of the sets {a,b} and {c,d} are {(a,c), (a,d), (b,c), (b,d)}. The truth table of C-functions for these pairs is:

Pair	C_1	C_2	C_3	C_4	C_5	C_6
(a,c)	0	0	0	0	1	0
(a,d)	1	0	1	1	1	1
(b,c)	0	0	0	0	0	0
(b,d)	1	0	1	1	0	1
TC	1	0	1	1	1	1

It shows that the subassemblies have contacts. Now, to check the global feasibility, consider the truth table of T-functions of the corresponding ordered pairs.

Pair	T_1	T_2	T_3	T_4	T_5	T_6
(a,c)	1	1	1	1	0	1
(a,d)	0	1	0	0	0	0
(b,c)	1	1	1	1	0	1
(b,d)	0	1	0	0	0	0
TT	0	1	0	0	0	0

The second entry of the resultant TT is a non zero entry, which indicates the disassembly direction of the subassembly (c,d) from the subassembly (a,d). So, the assembly direction of the subassembly (c,d) is the direction of 5 (complement of 2) or Y- direction.

The assembly sequences generated from the connectivity and precedence constraints are geometrically feasible sequences. But the resultant sequences may not always be mechanically feasible. To generate mechanically feasible sequences, stability constraints must also be taken into account. How the stability factors are taken into account to deduce mechanically feasible sequences is explained in Chapter 7.

The assembly sequence generation algorithm described above systematically generates all the geometrically feasible sequences. It is complete in the sense that it does not exclude any feasible sequence.

CHAPTER 6
ASSEMBLY SEQUENCE REPRESENTATION

6.1 INTRODUCTION

An assembly can have many different feasible assembly sequences. As it is practically difficult to represent each sequence individually, it is necessary to design a method to represent all these sequences in an efficient and compact manner. Choosing a suitable scheme for the representation of assembly sequences is an important decision both in creating an efficient assembly sequence planner and in designing an intelligent control system for the assembly process.

6.2 ASSEMBLY SEQUENCE REPRESENTATION SCHEMES

In the literature, several methods have been proposed for representing the assembly sequences. These representations can be classified into two groups: ordered lists and graphical representations.

6.2.1 Ordered Lists

Given an assembly that has N components, an ordered set of (N-1) assembly tasks $\{\tau_1, \tau_2, \ldots, \tau_{N-1}\}$ is an assembly sequence, if there are no two tasks that have common input subassembly, the output of the last task is the final assembly, and the input subassemblies to any task are either a single component or the output subassembly of preceding task. Such an assembly sequence can also be characterized by an ordered sequence of states in which the state s_1 is the state in which all parts separated (i.e., non-assembled), the state s_N is the state in which all parts are joined forming the whole assembly. An assembly sequence is said to be feasible if all of its assembly tasks and assembly states are feasible.

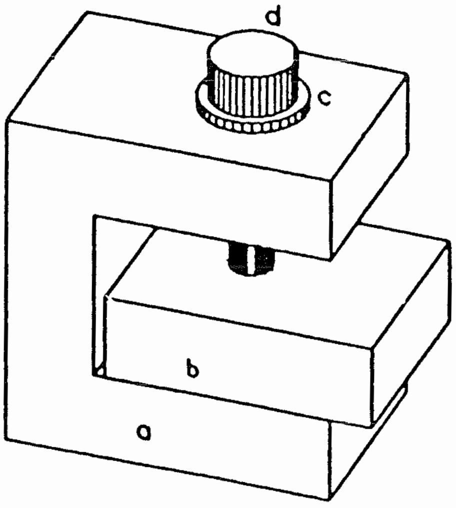
An assembly sequence, therefore, can be represented in several different ways (30) as described below.

- An ordered list of task representation. The number of elements in this list is equal to the number of parts minus one.
- An ordered list of binary vectors. Each vector must correspond to a state. The number of elements in this list is equal to number of parts.
- An ordered list of partitions of the set of parts. Each partition must correspond to a state. The number of elements in this list is equal to the number of parts.
- An ordered list of subsets of connections. The number of elements in this list is equal to the number of parts minus one.

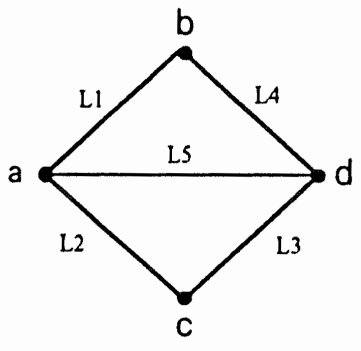
For example, a feasible assembly sequence for the assembly shown in Figure 6.1 could be represented as one of the following:

- A three-element list of task representations:

  ({{a},{b}}
  {{a,b},{c}}
  {{a,b,c},{d}})



(a) Assembly



(b) Liaison Diagram

Figure 6.1. An example assembly

- A four-element list of five-dimensional binary vectors:

  ([false, false, false, false, false]

  [true, false, false, false, false]

  [true, true, false, false, false]

  [true, true, true, true, true])

- A four-element list of partitions of the set of parts:

  ({{a},{b},{c},{d}}

  {{a,b},{c},{d}}

  {{a,b,c},{d}}

  {a,b,c,d})

- A Three-element list of sets of connections:

  ({c_1} {c_2} {c_3,c_4,c_5}).

Because each assembly sequence can be represented by ordered lists, it is possible to represent the set of all assembly sequences by a set of lists, each corresponding to a different assembly sequence. Although this set of lists might represent a complete and correct description of all feasible assembly sequences, it is not necessarily the most compact or most useful representation of the sequences. In particular, because many assembly sequences share common subsequences and common states, attempts have been made to represent sequences graphically which creates more compact representations that can encompass all feasible assembly sequences.

6.2.2 Graphical Representations

Several different diagrammatic representation schemes have been employed to represent the assembly sequences. The commonly used graphical methods to represent assembly sequences are:

- Precedence diagrams (56)
- State transition diagrams (74)
- Inverted trees (7)
- Liaison sequence graphs (16)
- AND/OR graphs (28)

6.2.2.1 Precedence Diagrams

One early attempt was the representation by a precedence diagram (56) that would actually encompass several possible assembly sequences. Assembly precedence diagram (APD) is a tree that encompasses all the possible assembly sequences. Development of an APD begins with listing of all individual assembly operations required to complete the assembly. Each operation is assigned a number and is represented by an appropriate circle in the diagram. The circles are connected by arrows showing the precedence relations. The APD is usually organized into columns. All the operations that can be carried out first are placed in the first column, and so on. Usually, one operation that appears in the first column is the placing of the base part in the fixture. Figure 6.2 shows an assembly precedence diagram for the assembly in Figure 6.1. The corresponding listing of operations is given in Table 6.1.

Table 6.1: List of assembly operations for the assembly shown in Figure 6.1

Operation No.	Operation description
1	Placing part a on to the fixture
2	Placing part b on part a
3	Placing part c on part a
4	Screwing part d through parts a, b and c

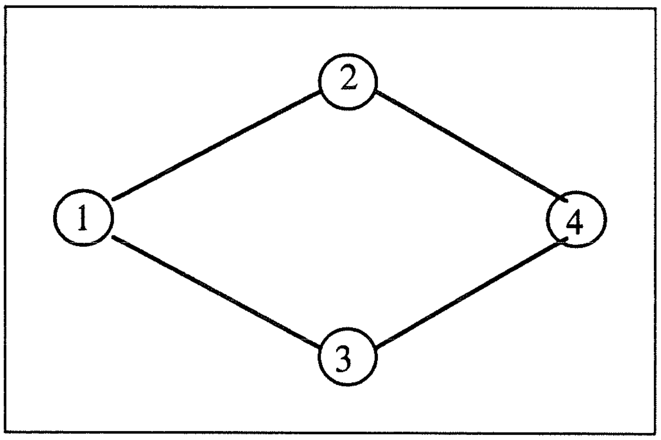


Figure 6.2. The APD for the assembly shown in Figure 6.1

In the diagram the following two sequences are represented: 1-2-3-4 and 1-3-2-4. The APD uses operational level detail about the assembly rather than simple contact or connection information and it also addresses the restrictions on the possible ordering of assembly. The APD has been used in assembly line balancing problems rather than representing assembly sequences. The shortcomings of the precedence diagram formalism are the lack of structure or algorithmic nature in the development of the diagram and that it is not a unique representation of the assembly. In other words, sometimes it does not show clear distinction between two different sequences.

6.2.2.2 State Transition Diagrams

Assembly state transition diagram (ASTD) representation (74) is based on the serial assembly of parts into a product. In its complete form, the ASTD represents all possible assembly sequences of a given group of parts. Figure 6.3 shows an example of an ASTD for the example assembly.

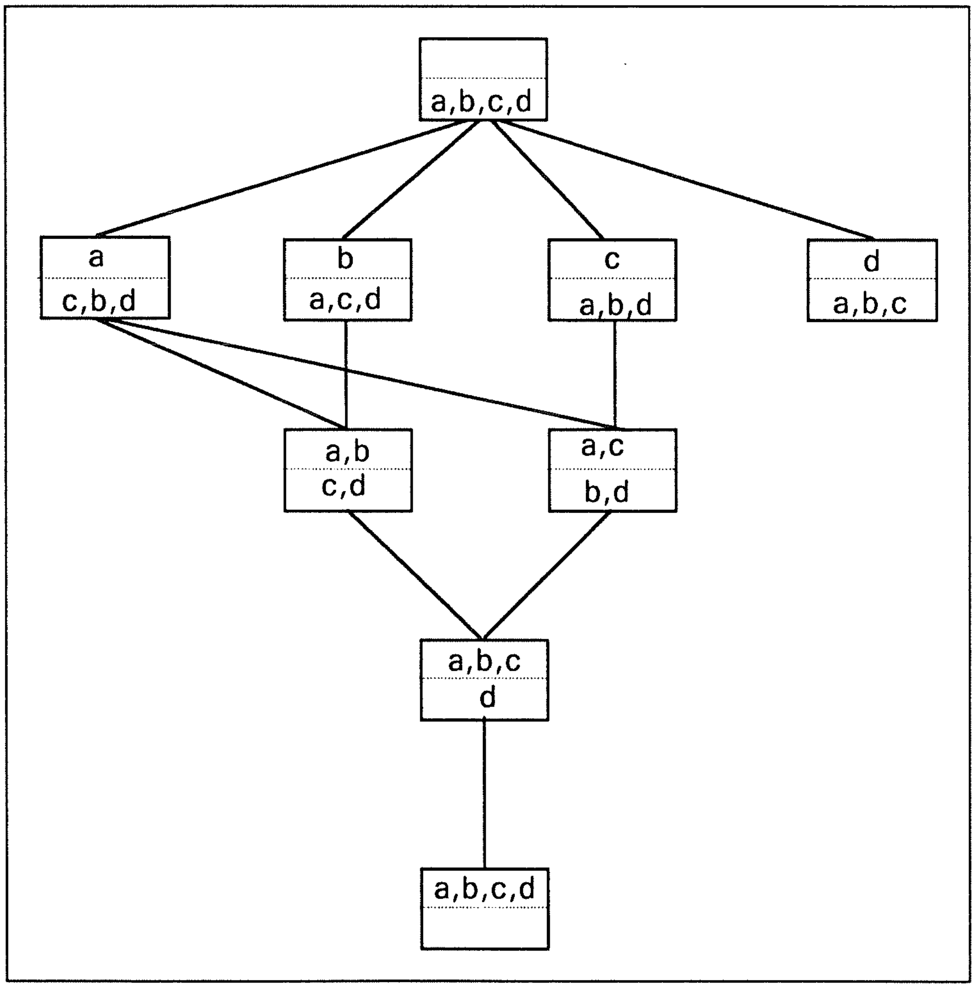


Figure 6.3. The ASTD for the assembly shown in Figure 6.1

In the Figure 6.3, the rectangles form the states. The arcs between the rectangles are the transitions. Each state represents the completed assembly of some particular parts. In the figure, the parts are represented by alphabets. The alphabets in the lower part of the rectangle represent the loose parts, whereas the alphabets in the upper part represent the assembled parts. Each transition between two states represents the assembly action of one distinct part.

This representation is based on the serial assembly of parts into a product. For sequences with subassemblies, each subassembly must be represented by a separate ASTD called child ASTD and a corresponding node in the parent ASTD. This representation is useful to represent the clusters of parts of the product. However, a single ASTD cannot provide a compact representation of all possible sequences.

Though the other three representation schemes have already been discussed in detail in Chapter 2, they will also be presented here and will be used for the same example assembly in order to have a comparative evaluation.

6.2.2.3 Inverted Trees

Bourjault (7) represented all the valid sequences in terms of an inverted tree which describes the possible orders of assembly. The complete inverted tree, representing all possible sequences for the example assembly is shown in Figure 6.4. This tree has 20 nodes and 17 arcs in it. The inverted tree gives the liaison sequences only, not the actual assembly sequences. Also, it does not contain any information about subassemblies.

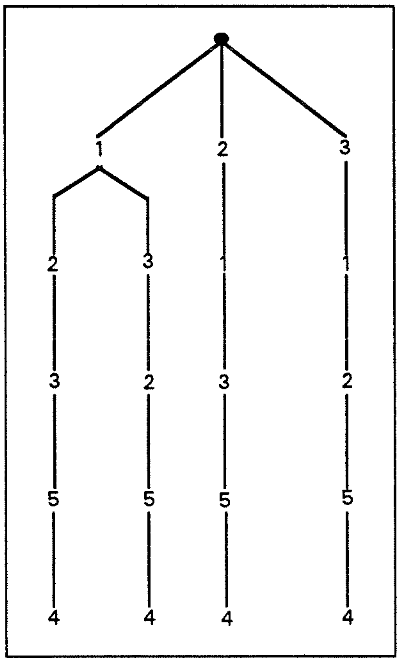


Figure 6.4. Inverted tree representation of liaison sequences for the assembly shown in Figure 6.1

6.2.2.4 Liaison Sequence Graphs

The liaison sequence graph (LSG) representation (16) is similar to that of state transition diagram, but this scheme gives liaison sequences instead of assembly sequences. However, here the states do not represent a set of parts, but a set of relations between the parts. The LSG is also referred to as directed graph of feasible liaison sequences. Figure 6.5 shows the LSG representation of sequences for the example assembly. This LSG consists of 9 nodes and 11 arcs.

6.2.2.5 AND/OR Graphs

Homem de Mello and Sanderson (28) have described an AND/OR graph representation of assembly sequences. Though the AND/OR graphs usually provide a

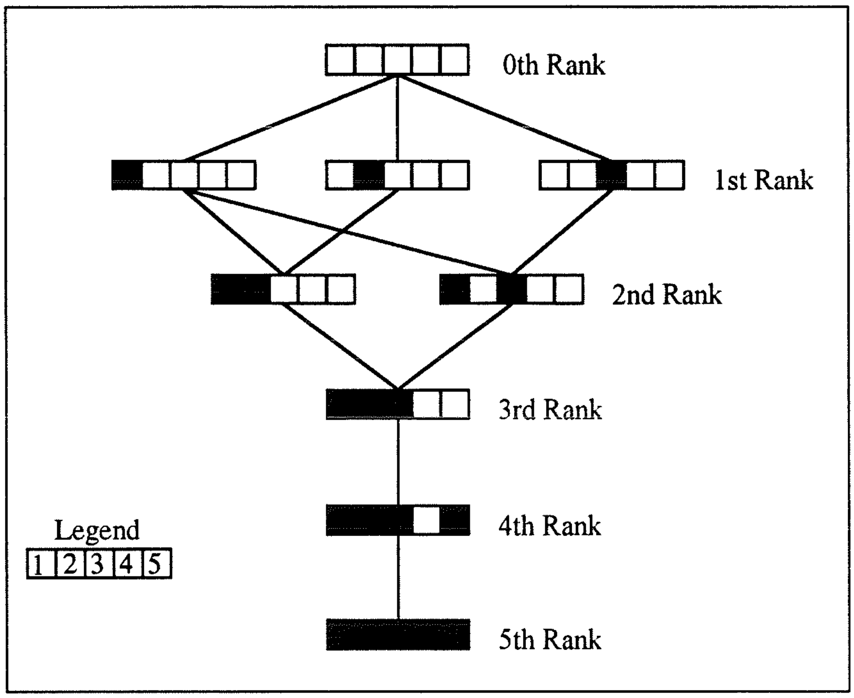


Figure 6.5. The LSG for the assembly shown in Figure 6.1

compact representation, they give the disassembly sequences only, and the assembly operations are not always the inverse of disassembly operations. Hence, this will hold only when each operation used in the disassembly is the reverse of a feasible assembly operation. The AND/OR graph of assembly sequences for the example assembly is shown in Figure 6.6. This AND/OR graph has 9 nodes and 7 hyperarcs which is less than that of the liaison models discussed above.

Normally when two sets of parts are mated, one set is moved while the other is held in place. None of the above representation schemes make this distinction for any of their operations. For example, adding part a to part b may not necessarily be the same as adding part b to part a. Especially in automatic assembly, feeding parts a and b are usually different from each other. Moreover, all the methods discussed above have their own shortcomings. So, there is a need to develop a new representation scheme that could give more details without losing the generality and completeness.

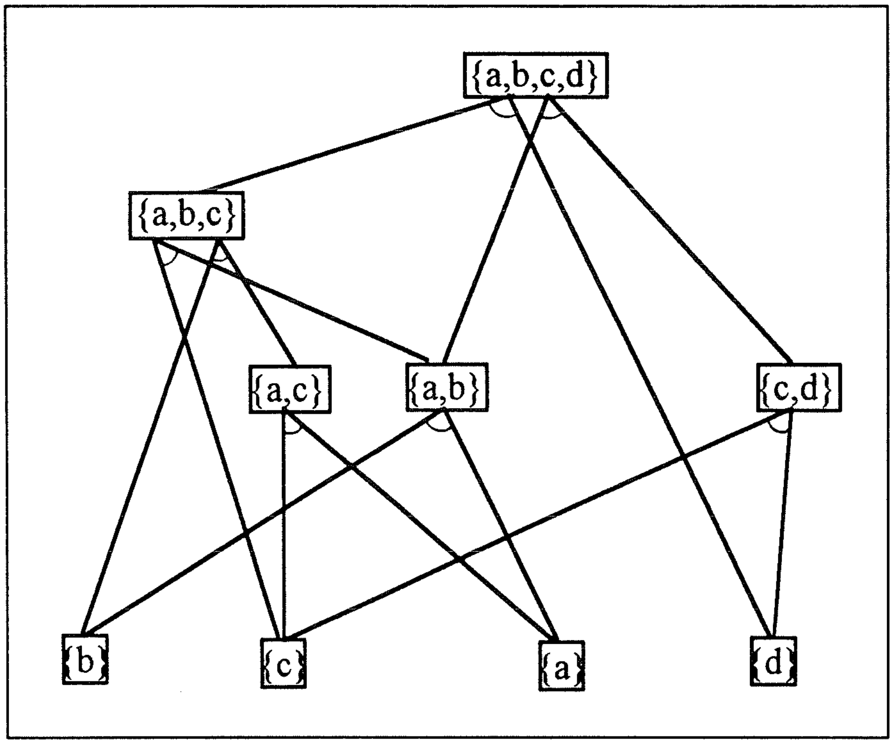


Figure 6.6. The AND/OR graph representation of assembly sequences for the assembly shown in Figure 6.1

6.3 ASSEMBLY SEQUENCE GRAPH (ASG)

To overcome the drawbacks of the sequence representation schemes discussed above, a new method of sequence representation called Assembly Sequence Graph (ASG) is proposed in this research. ASG has some common features with the LSG and AND/OR graph discussed above to retain the advantages of those schemes.

The nodes in the assembly sequence graph are the subsets of the parts set P, that characterize the feasible subassemblies. That means, each node corresponds to a subassembly. In the ASG, the nodes are represented by boxes and each box has N cells corresponding to the N parts in the assembly. A blank cell implies that the corresponding part is not assembled while a marked (or hatched) cell implies that the corresponding part has been assembled. At the top (first level), there are N boxes and each has one marked cell representing all the individual parts of the assembly in unassembled state. At the bottom (N^{th} level), there will only be one box with all the cells marked, which corresponds to the completed assembly. At level L, each box has L number of marked cells: that is, level L consists all the subassemblies having L number of components. Hence, level 1 represents the individual parts only and the subassemblies at level 2 consist of all topologically connected pairs. Similarly, the subassemblies at level 3 consist of three components each, and so on till level N. Each box (node) corresponds to an assembly state. But at level 1, all boxes correspond to only one state, i.e., the unassembled state. Lines connecting boxes represent the possible assembly state transitions.

Actually in ASG, one assembly task (joining of two subassemblies) has two arcs leading to the resultant subassembly, one from each of the constituent subassemblies. This pair of arcs can be referred to as a hyperarc or an and-arc corresponding to an assembly task. In other words, each hyperarc is an ordered pair in which the first element is a set of two nodes θ_i and θ_j and the second element corresponds to a feasible subassembly θ_k such that $\theta_k = \theta_i \cup \theta_j$, and assembly task characterized by θ_i and θ_j is feasible. Contrary to the AND/OR graph representation, the hyperarc in ASG is associated with the assembly of the two sets that correspond to the second element.

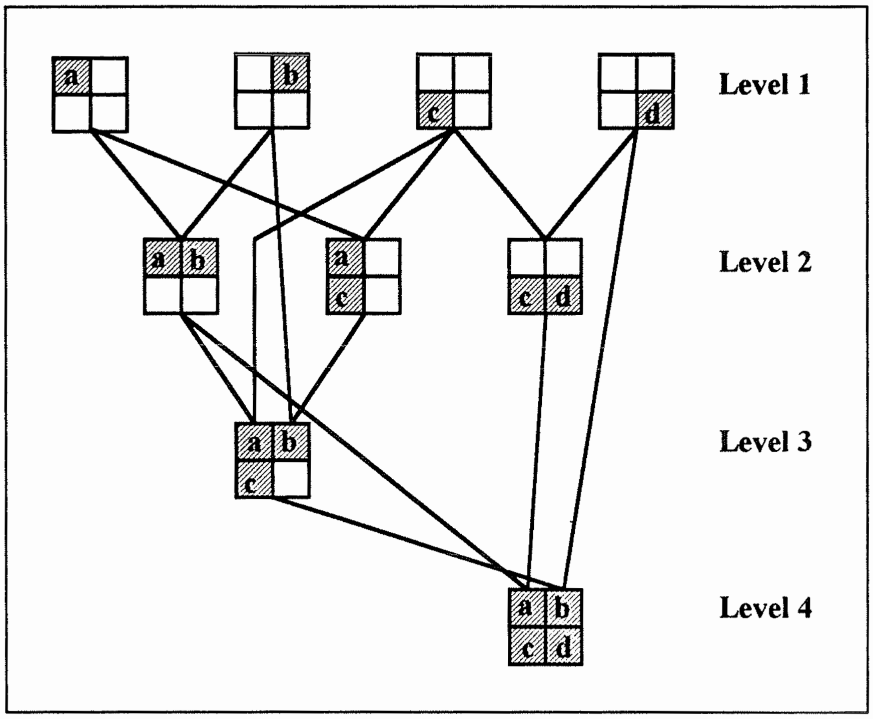
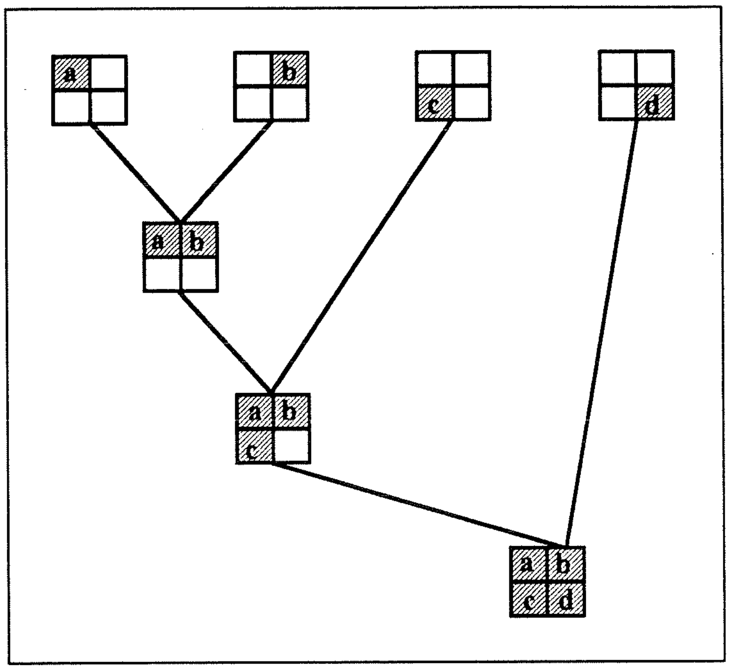
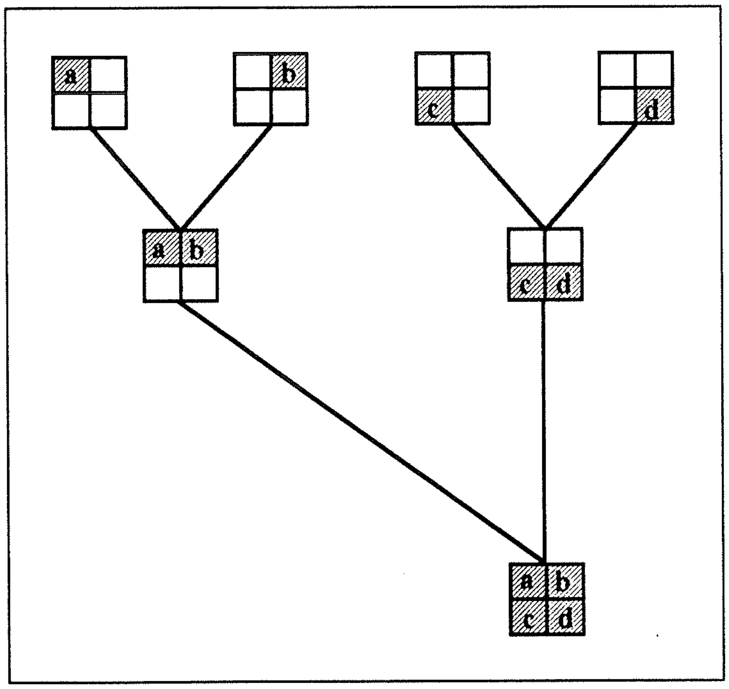


Figure 6.7. The ASG for the assembly shown in Figure 6.1

In the ASG, first level nodes (leaf nodes) and last level node (root node) are called primary nodes. These nodes will not be changed for the same set of components, even if there is a change in the relations among the components. The other nodes which are called secondary or intermediate nodes might be different for different relations and constraints. The first level nodes have only outgoing arcs, whereas the last level node has only incoming arcs. All the intermediate nodes must have at least one incoming hyperarc and one outgoing arc. Similarly, each hyperarc must be associated with two input nodes and one output node. Any secondary node without at least one input hyperarc and one output arc is known as a dead node. Similarly an hyperarc without two input nodes and one output node is considered as dead arc.



(a) Sequence 1



(b) Sequence 2

Figures 6.8. Two distinct sequence trees selected from the same ASG

As an example, Figure 6.7 shows the ASG for the feasible assembly sequences for the assembly shown in Figure 6.1. A path in the ASG representation is a connected subgraph in which each node has at most one hyperarc leaving from it, and at most one hyperarc incident to that node. The root node representing the final assembly has no arc leaving from it and the leaf nodes representing the individual components have no hyperarc leading to them. The ASG path whose root node is the bottom node of ASG and whose leaf nodes are the ASG leaf nodes is associated with an assembly plan and it is referred to as an assembly sequence tree. Figure 6.8 shows two distinct assembly sequence trees selected from the ASG shown in Figure 6.7. It can be observed that each of these trees gives one hierarchical representation of the assembly. It can also be seen that in all these trees, the number of assembly tasks and the number of nodes are the same irrespective of the structure of the tree.

6.3.1 Weighted Assembly Sequence Graph

The hyperarcs can be associated with a weight factor which could be the DOF of the subassembly or the assembly cost or time. This DOF can be directly obtained from the corresponding truth table of T-functions as discussed in Chapter 4. A subassembly or an assembly state is more stable, if it has lesser degrees of freedom within it.

Figure 6.9 shows an example of weighted assembly sequence graph. Here, the DOF of each assembly state is assigned to the corresponding hyperarc, if the subassembly is formed through the assembly task represented by that hyperarc.

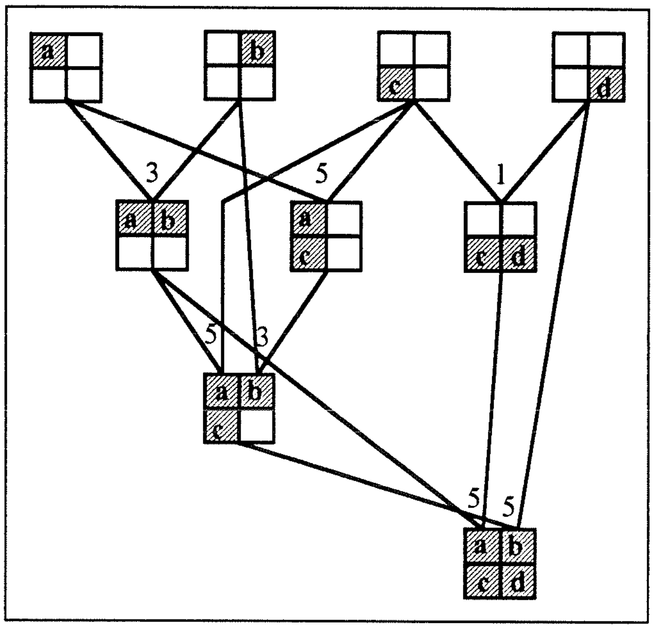


Figure 6.9. An example of a weighted ASG

Each line from one assembly state to the other can also be assigned an attribute such as assembly task time, cost etc. which are very useful for the evaluation of assembly plan. Since estimating the cost for each assembly task involves enormous amount of computation, the assembly costs are assigned to the arcs in the ASG after pruning the graph by eliminating all unwanted assembly tasks and assembly states. The pruning of the ASG will be explained in the next chapter.

6.3.2 ASG Generation Algorithm

The procedure to generate the ASG of all feasible sequences is described below. This procedure can be made more efficient by linking the evaluation of stability of each subassembly.

```
Step 1: Initialization
            Set of Components $P = P_1, P_2,\ldots,P_N$
            Level = L
            Number of boxes in level $L = S_L$
            box i of level j = BOX[i,j]
Step 2: First level, L = 1
            $S_1 = N$
            Draw N boxes, each box with (N-1) empty cells and one filled cell
            uniquely representing each component of the assembly
Step 3: Subsequent levels, L = L + 1
            $S_L = 0$
            FOR X = 1 TO ABS(L/2)
              IF ($S_{L-X}$ = 0 ) OR ($S_X$ = 0) THEN NEXT X
              FOR I = 1 TO $S_{L-X}$
                  BOX1 = BOX[I,L-X]
                  FOR K = 1 TO $S_X$
                    BOX2 = BOX[K,X]
                    IF (BOX1 $\cap$ BOX2) $\neq$ 0 THEN NEXT K
                    ELSE IF  (BOX1 $\cup$ BOX2) = "infeasible" THEN NEXT K
                      ELSE DRAW NEWBOX with N cells at Level L
                          FILL the corresponding cells
                          DRAW an hyperarc from BOX1 and BOX2 to NEWBOX
                          $S_L = S_L + 1$
                  NEXT K
              NEXT I
NEXT X
IF L = N THEN
          DELETE all the dead nodes and the corresponding arcs
          STOP
ELSE GO TO STEP 3
```

6.3.3 ASG Construction

To explain the construction of an ASG, consider the example assembly shown in Figure 6.1. The steps required for the construction of an ASG based on the above algorithm can be explained as follows (Figure 6.10).

This assembly has four components a, b, c and d, therefore N = 4. In the first level, there are four boxes with four cells each, and one cell is marked to correspond to one component of the assembly. The first cell in the first box is marked as "a" and similarly other boxes corresponding to the individual components in the unassembled state. To make it clear, the marked cells are further highlighted by hatching.

Level 2 consist all the topologically connected pairs. Based on the procedure described in the previous chapter, the feasible two part subassemblies are (a,b), (a,c), (a,d), (b,d), and (c,d). So this level will have 5 boxes, one box for one subassembly, and each box with two marked cells corresponding to two components of that subassembly. Each node has one or more input hyperarcs representing the assembly tasks through which the subassembly is formed.

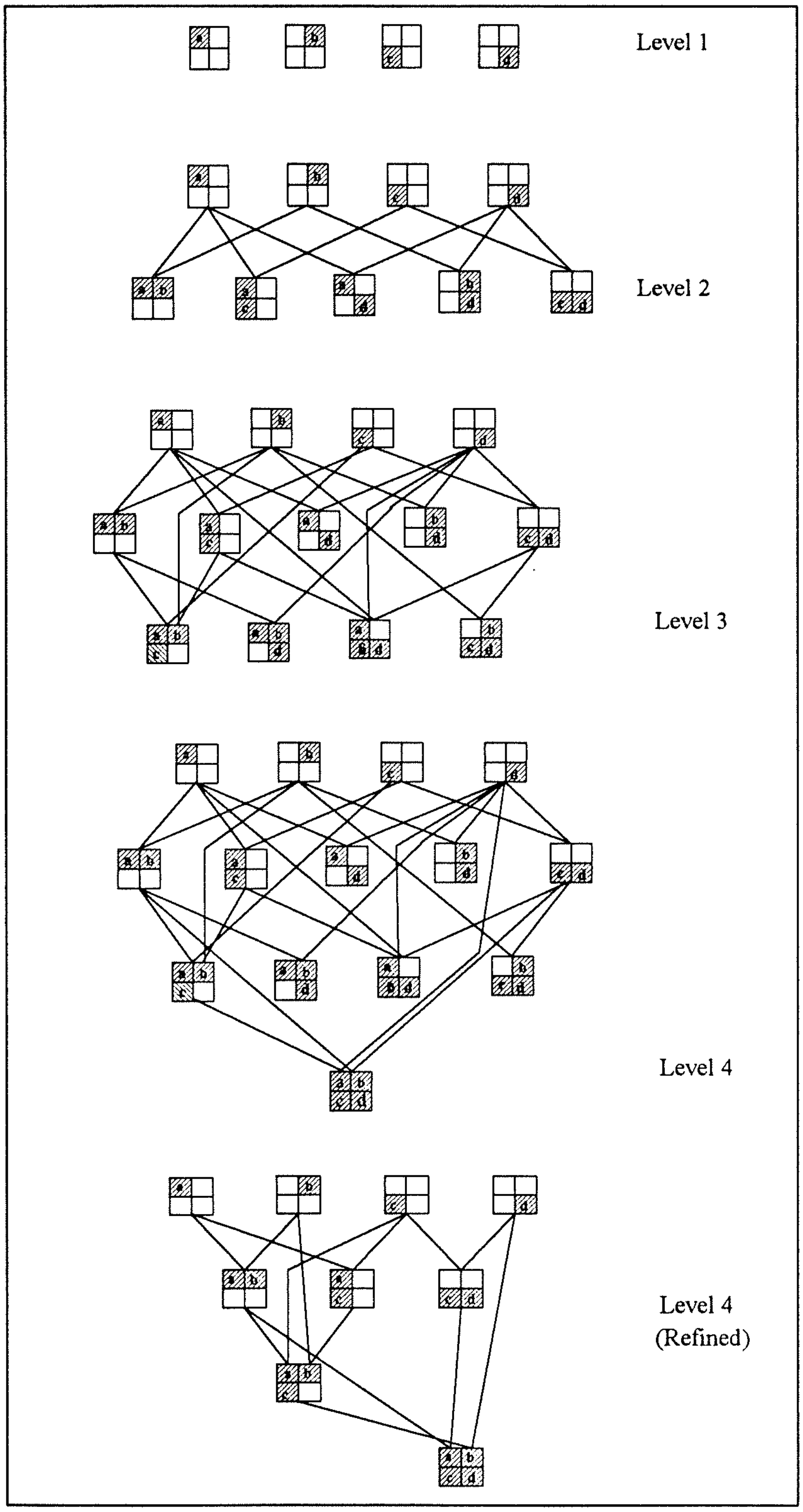


Figure 6.10. Construction of an ASG

Similarly level 3 has 4 boxes with three marked cells corresponding to the subassemblies (a,b,c), (a,b,d), (a,c,d) and (b,c,d). In the last level (fourth level), there is only one box with all the cells marked, representing the whole assembly.

Note that there are no arcs leaving from the boxes that correspond to (a,d) and (b,d) at level 2 and from the boxes that correspond to (a,b,d), (a,c,d) and (b,c,d). These nodes are dead nodes and no higher order assemblies are feasible from these subassemblies. Now, all the dead nodes and the corresponding hyperarcs are deleted from the graph. The resulting graph is the compact representation of all the feasible assembly sequences. An assembly plan is a path from one of the boxes at first level leading to the box in the last level through a series of edges in the graph. For example {(a)-(a,b)-(a,b,c)-(a,b,c,d)}is a feasible assembly sequence.

6.3.4 Complexity of ASG

The amount of computation involved in the generation of the ASG for a given assembly depends on the number N of components that make up the assembly, on how these components are interconnected and also on the resulting ASG.

The number of prospective combinations that must be analyzed will be used as a measure of the computation involved in the generation of all feasible assembly sequences. The computational complexity of the proposed assembly sequence generation algorithm can be expressed in terms of the number of nodes and the number of arcs in the ASG.

For an assembly of N components, the number of nodes in the first level is N and in the last level this number is equal to 1. The number of nodes in the intermediate levels of ASG depends on the number of feasible assembly sequences. The maximum possible number (S_L) of nodes in any level can be estimated as

$$S_L = {}_NC_L = \frac{N!}{L!*(N-L)!}$$

where L is the level in the ASG

The maximum limit ($S_{\max}$) for the number of nodes in the ASG can be obtained by simple summation of the number of nodes at each level. Therefore,

$$S_{\max} = \frac{N!}{1!*(N-1)!} + \frac{N!}{2!*(N-2)!} + \ldots + \frac{N!}{(N-1)!*(N-(N-1)!} + \frac{N!}{N!*(N-N)!}$$

$$= N + \frac{N!}{2!*(N-2)!} + \ldots + \frac{N!}{(N-1)!} + 1$$

or

$$S_{\max} = \sum_{L=1}^{N} S_L = \sum_{L=1}^{N} \frac{N!}{L!*(N-L)!}$$

Since there must be at least (N-2) intermediate states (secondary nodes) to complete the assembly, the minimum possible number of intermediate states is (N-2). Therefore, the lower bound ($S_{\min}$) for the number of nodes in an ASG is the sum of (N-2) and primary nodes (N+1).

$$S_{\min} = N + 1 + N - 2$$

$$= 2N - 1$$

To estimate the number of arcs in the ASG, the number of prospective combinations that must be analyzed for each 2 component subassembly is $(2^{2-1}-1)$, for each 3 component subassembly, it is $(2^{3-1}-1)$, for each L component subassembly, this number is $(2^{L-1}-1)$ and so on. The maximum number of hyperarcs at any level L (A_L) is equal to the number of nodes in that level multiplied by the number of possible combinations to form each of these nodes. This can be written as,

$$A_L = S_L * (2^{L-1}-1)$$

$$= {}_NC_L * (2^{L-1}-1)$$

$$= \frac{N!}{L! * (N-L)!} * (2^{L-1}-1)$$

The upper limit for the number of hyperarcs (A_{max}) can be obtained by adding these numbers for each level. Therefore,

$$A_{\max} = {}_NC_2 * (2^{2-1}-1) + {}_NC_3 * (2^{3-1}-1) + \ldots + {}_NC_N * (2^{N-1}-1)$$

$$= \sum_{L=1}^{N} {}_NC_L * (2^{L-1}-1)$$

$$= \frac{3^N+1}{2} - 2^N$$

Since it is assumed that each assembly task joins two subassemblies, the number of assembly tasks in an assembly plan is equal to the number of parts in the assembly minus one, i.e. (N-1). Hence, the minimum number of hyperarcs (A_{min}) in the ASG is

$$A_{\min} = N - 1$$

In the above discussion, the upper bounds are for the assemblies in which every part can be assembled to the other part and the lower bounds are for the assemblies with only one feasible sequence.

Table 6.2. The number of nodes and arcs in the ASG as a function of number of parts

Number of Parts (N)	Number of Nodes		Number of Arcs	
	Upper Limit (S_{max})	Lower Limit (S_{min})	Upper Limit (A_{max})	Lower Limit (A_{min})
2	3	3	1	1
3	7	5	6	2
4	15	7	25	3
5	31	9	90	4
6	63	11	301	5
7	127	13	966	6
8	255	15	3025	7
9	511	17	9330	8
10	1023	19	28,501	9
15	32,767	29	7,141,686	14
20	1,048,575	39	1,742,343,625	19

Based on the equations derived above, the number of nodes and the number of prospective combinations through which these nodes can be formed are shown in the Table 6.2 as a function of the number of parts that make up the assembly. The entries in this table are given as a reference since it is very unlikely that there would be a fifteen part assembly in which every part is connected to the other part.

The computational complexity of the algorithm can be reduced by imposing some of the constraints which will be discussed in the next chapter, before generating the sequences. The algorithm generates only sequences which do not violate the prescribed constraints.

6.4 ASSEMBLY SEQUENCE TABLE

Although an ASG provides a complete and compact representation, it becomes too clumsy for higher number of components. An alternative scheme is proposed that describes the assembly sequence graph clearly. In this scheme, all the assembly states and tasks represented in the ASG will be listed in the form of a table called Assembly Sequence Table (AST). The AST is exactly equivalent to the ASG. The equivalent AST for the ASG shown in Figure 6.7 is given in Table 6.3.

Table 6.3. The AST equivalent to the ASG shown in Figure 6.7

Level	Nodes (States)	Arcs (Tasks)
2	{a,b}	{(a), (b)}
	{a,c}	{(a), (c)}
	{c,d}	{(c), (d)}
3	{a,b,c}	{(a,b), (c)}; {(a,c), (b)}
4	{a,b,c,d}	{(a,b,c), (d)}; {(a,b), (c,d)}

In the AST, the first column indicates the level number in the ASG. The first level nodes are not included in the AST, since they are not the result of any assembly tasks. The second column and third column in the AST correspond to the nodes and arcs of the ASG respectively.

In the estimation of bounds for the number of nodes, the number of first level nodes is fixed in both the cases and further these nodes are not included in the AST. So, the bounds for the number of states in an AST can be specified without including the first level nodes. The limits for the number of tasks in the AST are same as the number of arcs in the ASG, whereas the limits for the number of states in the AST are:

$$S_{\max} = \sum_{L=1}^{N} \left(\frac{N!}{L!*(N-L)!} \right) - N$$

$$= \sum_{L=2}^{N} \frac{N!}{L!*(N-L)!}$$

$$S_{\min} = N - 1$$

With these new values, the lower limits for the number of nodes and arcs are the same. That means, each state can be formed by only one task.

The AST describes how each assembly state is formed and the constituent subassemblies. Since the tasks in the AST do not carry weightage factors such as assembly time, cost, etc., it is considered to be less efficient compared to the ASG to perform evaluation of sequences.

Together, the AST and ASG provide an efficient representation of assembly sequences. The AST or the ASG is complete in that it includes all feasible assembly sequences and they can be used as the basis for searching for the best solution. They do not show any state more than once. It is also possible to show a clear distinction between assembling part i to part j and assembling part j to part i by treating the hyperarc as two individual arcs.

CHAPTER 7
SELECTION AND EVALUATION OF ASSEMBLY SEQUENCES

7.1 INTRODUCTION

The assembly planning algorithms described in the previous chapters to generate the set of all feasible assembly sequences, have focused almost exclusively on hard constraints imposed by the geometric limitations of the constituent components and the product itself. Under some circumstances, the engineer may require to impose some additional constraints or conditions upon the assembly sequences in addition to the size and geometry expressed in the precedence relations. The next level of assembly constraints that can further limit the available sequence alternatives are the strategic constraints or soft constraints.

The choice of assembly sequence affects many factors such as difficulty of assembly operation, fixturing requirements, potential damage of parts during assembly, ability to do inprocess testing, characteristics of end product, cost of assembly, etc. The impact of the assembly sequence on the entire manufacturing system is enormous. Whatever the approach, a typical product can have several hundreds or thousands of feasible assembly sequences and this number increases exponentially when parts count increases. Obviously, it is very difficult to analyze all these alternatives without some efficient way to enumerate them. For each assembly plan, there are several possible assembly systems to simulate before the practical implementation and it is practically impossible to analyze all the feasible sequences. Therefore, there is a need to develop some procedures to reduce the sequence count to select the best assembly sequence that most nearly meets our needs for a particular purpose within the available resources.

A variety of criteria can be used to choose the required assembly plan that will be used in the actual assembly process. Several researchers followed different approaches to apply these criteria for reducing the sequence count to a practically convenient level. For example, Wolter (75) presented some criteria based on manipulability, on fixture complexity, and on the number of different directions from which operations are performed. Klein (35) addressed other criteria, including the minimization of reorientations, fixturing requirements and facility constraints. In Baldwin et al (1), evaluation and selection are done by manual editing via computer aided instructions and criteria options.

Henrioud et al (25) proposed three sets of criteria: operational complexity, logistical complexity, and strategic advantages. They stored these constraints into a database, and this database is searched for each sequence in the inverted tree, in order to find whether the sequence is acceptable or not. To consider several incompatible or contradictory strategies, the method has to run several times with each strategic constraint i order to obtain different sets of assembly trees corresponding to different strategies.

Homem de Mello and Sanderson (29) introduced two criteria: maximizing the number of sequences in which the assembly task can be performed and minimizing the total assembly time through a parallel execution of assembly tasks. The selection of the best sequence was seen as a search problem in the AND/OR graph. Barakat and Vallet (2) developed an algorithm to evaluate the assembly sequences based on the assembly system related constraints. Bonschancher and Heemskerk (4) introduced a technique that reduces the complexity of sequence planning by grouping parts into clusters.

Reddy and Ghosh (58) described various soft constraints to be considered in the selection and evaluation of assembly sequences. This chapter describes some of these factors and how they are implemented in the selection and evaluation process by providing some editing features in the ASG.

7.2 SELECTION AND EVALUATION CRITERIA

Many different strategic factors (soft constraints) which are independent of the geometry of the product influence the choice of the assembly sequence. The criteria for selection are not clearly defined a priori, but are specific to the product under study and also depend on the facility requirements. The selection and evaluation criteria are judgmental factors discussed in this chapter are grouped into two major categories.

- Qualitative factors
- Quantitative factors

7.2.1 Qualitative Factors

These factors are qualitative in nature and they will be used for qualitative analysis of sequences to reduce the sequence count to a few candidate sequences. These criteria pertain to characteristics or attributes of particular assembly states or state transitions (assembly tasks). Particular states of assembly can be either desirable or undesirable from a manufacturing standpoint, and these can be usefully applied in sequence selection. The following are the most commonly considered criteria for the selection of sequences.

Subassembly Stability: One important qualitative attribute is the stability of the subassembly or part in a particular state or during a state transition. Briefly, stable subassemblies are desirable states and unstable subassemblies are best avoided if possible in that they call for the complication and expense of stabilizing jigging. The sequences that are generated by considering the hard constraints are the geometrically feasible sequences only. The application of the stability factor will result in mechanically feasible sequences. For the most part, a stable assembly state is desirable, while an assembly state having unstable components is best avoided, if at all possible. However, part or assembly stability is more than a question of whether the part is mechanically fastened or held in some way to the rest of the assembly.

The issue of part or assembly stability is important in that unstable components may require fixturing or jigging at some points of the assembly process to maintain their location. While in some cases there is no alternative to temporary fixturing, some assembly sequences can eliminate the need for special jigging.

Specified order of a component or subassembly: This constraint represents an arbitrary partial assignment of sequence appropriate to situations where it is known that the particular partial sequence is the best among the others, or where it is known that no advantage is available from any of the other partial assembly sequences for a set of parts. Sometimes, it is desirable to add a particular component in a specific order. For example, it is often necessary to place expensive parts toward the end of the assembly process. Inversely, it is often best to place heavy parts as soon as possible. Usually, a heavy, or large part is chosen as the base part. Another way of selecting the base part is the one that has connectivity with more parts. Certain orders of assembly can reduce the skill level or dexterity required by the operator or degrees of freedom required of automation to complete the assembly task.

Clustering of components: Clusters are groups of components which are assembled separately prior to being attached to the major assembly or base part. By clustering the components, we can reduce the number of components artificially in the final assembly. Many products have natural subassemblies that arise as a result of modular design as well as because of manufacturing advantages. Clustering of components sacrifices the completeness if it is done before the generation of sequences. But if it is applied during the selection phase, it does not cause any loss of completeness. Moreover, for most of the products with large parts count, this loss of completeness is not a serious limitation because those natural subassemblies are assembled independently anyway.

The sequence generation algorithm generates a large number of potential or candidate subassemblies, particularly with assemblies of significant parts count. Many of the generated subassembly candidates, however, are of little value or are undesirable. The nonfunctional subassemblies must be eliminated as potential sequence alternatives.

Fixture complexity: The question of fixturing or jigging is another interesting factor to be considered in the sequence selection process. Fixtures are used in industrial assembly for various applications: to support a base part, to stabilize an unstable subassembly, etc. The sequence must be selected so that the partially built assemblies hold themselves together as much as possible without requiring any temporary fixturing. From a cost standpoint, temporary fixturing creates additional tooling expense to the cost of product assembly. Temporary support or stabilizing the fixtures adds no value to the product during the assembly process, and therefore consume labor time as well as machine time which could be better applied to other assembly tasks.

Uniform assembly direction in successive tasks: In robotic assembly, it is always preferable to insert all parts, as much as possible, from a single direction. This requires less robot motion, simplifies fixturing requirements, eliminates flip-overs, requires a less dexterous robot and avoids extra operations to reorient the subassembly. Savings in both set-up and run-time costs can be realized under this criterion.

Modularity: There are several reasons to introduce some modularity in the assembly sequence by making the assembly with subassemblies. This criterion maximizes the amount of parallelism that is possible in the execution of the tasks. It is also useful in order to minimize the total execution time. The simultaneous execution of assembly tasks, in most cases, reduces the total assembly time. Another major reason for including subassemblies as part of the assembly sequence is the ability to functionally test or inspect the subassemblies prior to their attachment to other components. This ability to inspect or test prior to installation can have considerable impact on final repair costs of the assembled product.

Linear assembly sequences will require a linear disposition of the assembly equipment. The operational reliability is likely to be low for an automated assembly line which is quite long (about 30 stations or more). Use of subassemblies will allow us to shorten each of the sections of the assembly line.

For products of the same family, certain subassemblies could be common to all the members of the family. It may become economically justifiable to maintain an inventory of these subassemblies. Subassemblies could be so constituted that they can be bought from a specialized supplier. In addition to that, replaceable components can be constituted as subassemblies. Such subassemblies could be replaced periodically during maintenance.

Proximity: For large assemblies, it is preferable to perform consecutive operations in locations that are physically close to each other. For example, if a riveter is working on an airplane, he would prefer not to insert alternate rivets into alternate wing-tips. This speeds the execution time for the assembly process.

Manipulability: One might wish to perform more difficult operations with more easily handled parts. For example, to attach a bolt to an engine block, it is easier to fixture the engine block while screwing in the bolt, rather than fixturing the bolt while screwing on the engine block.

Accessibility: The viability of an assembly task can also be characterized by the ease of assembly or the physical accessibility to complete the task effectively. The poor accessibility could result in damage of some assembled components.

Tool changes: The sequences that require frequent tool changes can result in increased tooling cost. The activities should be sequenced so that, operations with similar tool requirements are grouped together, so that a minimal number of tool changes are required. This saves time during the execution of the assembly plan.

Stacking Constraints: These constraints are caused by external fasteners: screws, for instance. When these special parts hold together a stack of other components, it is usually best to impose a given assembly sequence for this group of parts.

Some of the criteria listed here are quite expensive to optimize. For example, optimizing a plan for locality requires, essentially, the solution of the traveling salesman problem. In most cases, it would suffice to find a near optimal solution, and this is generally quite feasible.

The relative importance of the criteria depends on the method of assembly being used. For example, if a single robot is working on a large assembly, proximity or physical location is quite important. But on an assembly line, where consecutive operations are performed by different robots, location may have no importance at all. Directionality is important in a workcell, but is less important on an assembly line. Minimizing tool changes is also less important in a workcell than on an assembly line, since consecutive operations are likely to be performed by different machines in the latter case. Clearly, the criteria may contradict each other. Because of this, the planner must be able to reach compromises between conflicting criteria. Thus, some type of weights must be given to the criteria to indicate their relative importance. Moreover, some of these criteria are user selectable while others are not.

7.2.2 Quantitative Factors

Despite using the criteria listed above to do winnowing, there may still be many sequences to be evaluated. To allow ease of choice, automatic evaluation of the selected assembly plans must be performed to determine the final choice. The quantitative criteria employ more concrete characteristics about assembly processes. These factors are most often associated with the attributes that directly influence the assembly cost. The quantitative characterizations may include times to accomplish the assembly tasks, costs of the hardware required, costs of fixturing or tooling needed to secure unstable states, and so forth. The user can apply a simple criterion to the choice of assembly sequence, such as shortest time or cost path through the weighted assembly sequence graph. The commonly used evaluation factors are:

- Assembly time
- Assembly cost
- The number of part reorientations during assembly
- The number of fixtures or tools required
- The number of assembly stations
- Degree of difficulty of assembly task, etc.

Productive use of available assembly time is a key to efficient, cost-effective assembly. While it is impossible to completely avoid all non-value added operations, it is desirable to select assembly sequences which minimize the number of non-value added tasks.

Inefficiencies also arise from reorientations of the assembly and frequent tool changes. Again, not all reorientations or tool changes can be avoided, as some are required to provide necessary access to a part of assembly. It is desirable, however, to utilize assembly sequences which minimize them, if at all possible. Assembly reorientations or flips which occur in a particular station also consume productive assembly time as, most often, no other operations can take place during the change of position.

Resource utilization is also very important in the evaluation of sequences. It is always preferable to accomplish as many tasks as possible in the available cycle time so as to reduce the required number of resources (or workstations) and thus, the capital expenditures associated with the assembly of the product.

7.3 QUALITATIVE ANALYSIS OF SEQUENCES

As discussed above, determination of the best assembly sequence can be judgmental, qualitative and quantitative or combination of these. In this work, the selection of sequences from the ASG used the similar lines of winnowing discussed by Nevins et al (52). This editing process could be done by either deleting the unwanted assembly states and unwanted assembly tasks or retaining the most desirable assembly states and tasks while deleting the others as described below.

1. Eliminate unacceptable assembly states, by deleting the corresponding nodes or boxes from the ASG. Then remove all the arcs that are leaving or entering these nodes and also the resulting dead nodes.

   For example, from the ASG shown in Figure 6.7, if the subassembly {a,c} is not desirable for some reason, then the corresponding node must be deleted. When this is done, the arc through which the node {a,c} was formed will be left without an output node and the arc that joins the nodes {a,c} and {b} will be left without one of the input nodes. As these two arcs now are dead arcs, hence they should also be removed from the ASG. The resulting ASG is shown in Figure 7.1

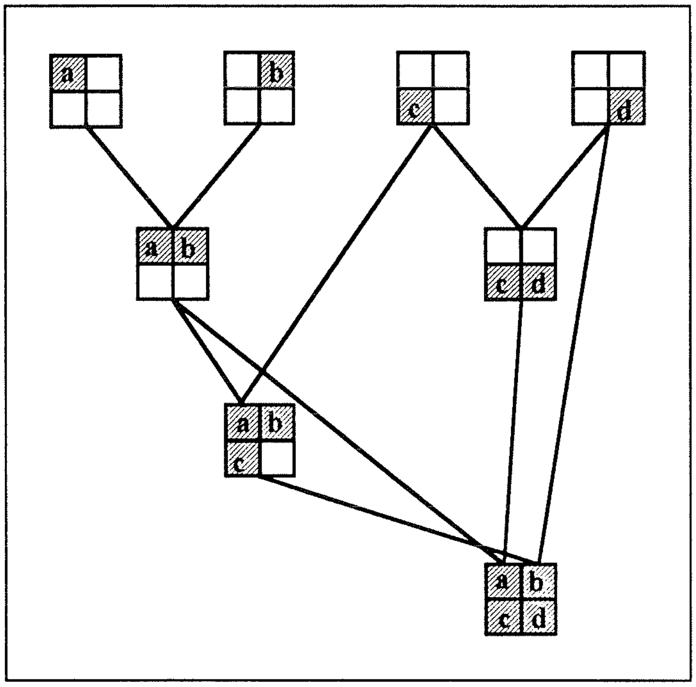


Figure 7.1. The ASG after deleting unwanted assembly state {a,c} form the ASG shown in Figure 6.7

2. Eliminate unacceptable assembly tasks, by deleting the corresponding arcs from ASG. Then remove all the nodes (except first and last level nodes) with no arcs leaving or entering.

For example, suppose the assembly task of joining the subassembly {c,d} to the subassembly {a,b} is considered as a difficult assembly task, because it requires a flip-over to complete the assembly. So, the corresponding arc {(a,b), (c,d)} must be deleted from the ASG shown in Figure 7.1. This results in the node {c,d} without any outgoing arcs, hence this is a dead node. Again, deletion of this node leads to the deletion of the arcs that generate this node. After these modifications, the resulting ASG will be as shown in Figure 7.2.

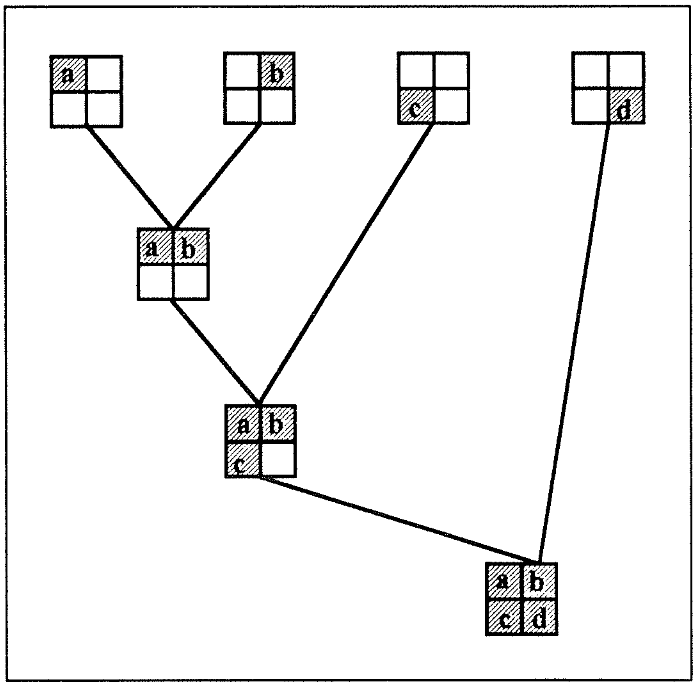


Figure 7.2. The ASG after deleting unwanted assembly task {(a,b), (c,d)} from the ASG shown in Figure 7.1

It can also be observed that, if the same consideration is applied before deleting the assembly state {a,c} in the Figure 6.7, the resulting ASG would be as shown in Figure 7.3. This ASG is different from the ASG depicted in Figure 7.2.

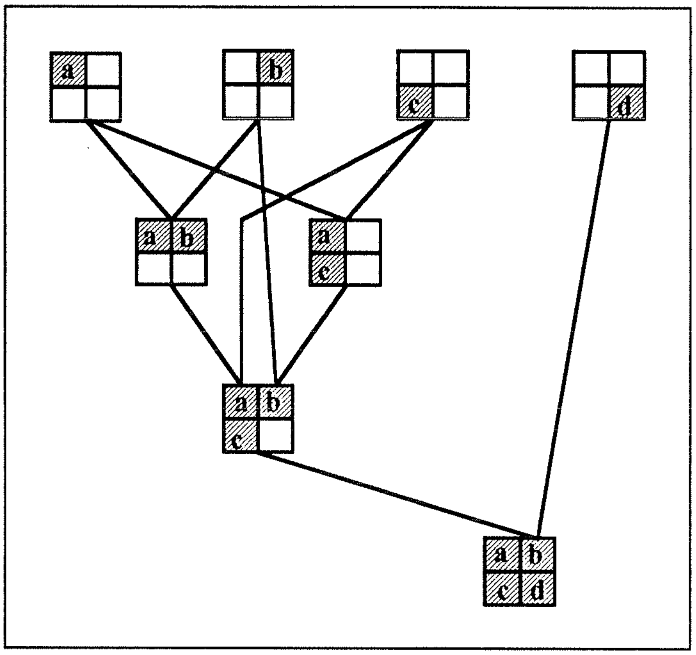


Figure 7.3. The ASG after deleting the assembly task {(a,b), (c,d)} from the ASG shown in Figure 6.7

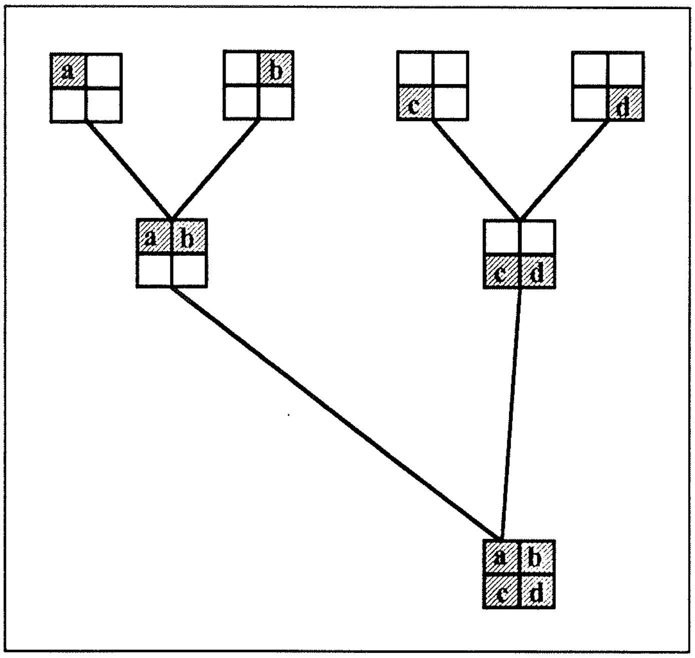


Figure 7.4. The ASG retaining the sequences with subassemblies only

3. Retaining the desirable assembly tasks and states will also be performed on the similar lines. For example, if the modularity is the primary concern, the sequences with subassemblies {a,b} and {c,d} must be retained. After removing all other states (nodes) and tasks (arcs), the resulting ASG is shown in Figure 7.4.

It can also be observed that if the assembly state {a,c} is deleted from the Figure 7.3, the resulting ASG would be the same as the one shown in Figure 7.2. This shows that irrespective of the order in which selection criteria is applied, the net result would be the same.

Editing the AST is exactly the same as that of the ASG. Here, to eliminate the unwanted assembly states, one deletes the entire row of that state from the table. There might be some assembly tasks which contain the deleted assembly state as its constituent. Hence, assembly task should also be deleted from the right hand side column of the table. The deletion of an assembly task might leave some dead states in its previous levels, which must also be deleted. This process continues until there are no dead states and dead tasks in the table.

Pruning the sequences from an ASG is very convenient as it gives visual representation of all assembly states and tasks. All the nodes or arcs that are associated to the deleted arcs or nodes can be visualized and deleted from the graph. But for assemblies with more parts, usually the number of nodes and arcs are quite large, which makes the ASG more clumsy. Editing of an AST is more convenient especially when there are many assembly states and tasks, but it requires a rigorous search process as it does not provide any visual representation. Moreover the assembly tasks represented in the AST do not carry any weightage factors and so the AST is not suitable for quantitative evaluation of the sequences. So, for assemblies with larger parts counts, it is preferable to start the editing process using the AST until it becomes reasonably less complex, then continue with the ASG.

It is also observed that, the order of application of the constraints is also very important. Even for the same constraints, if they are applied in a different order, the resulting ASG (or AST) will be different and also the resulting number of sequences. Hence care must be taken in implementing these criteria.

7.4 QUANTITATIVE ANALYSIS OF SEQUENCES

The qualitative characterization discussed above will result in a few (usually four or five) candidate sequences for further analysis. The next step is the quantitative evaluation of the sequences to determine the final choice. The quantitative factors that are considered in this research for the evaluation of assembly sequences are assembly cost and assembly time. The assembly cost or time for each assembly task will be assigned to the corresponding hyperarc in the ASG. Then the final sequence will be the assembly tree with minimum total assembly cost or time. The assembly costs or times are estimated using the DFA Toolkit (79). This software takes the description of all individual components and a specific sequence, and automatically gives the total assembly cost. It estimates the assembly cost based on the shape, size, and other features of the parts, and the degree of difficulty of the assembly tasks.

Considering the above example, the qualitative selection process produced only one sequence which eliminates the necessity of quantitative evaluation process. But to explain how the quantitative evaluation is performed, all three sequences with the associated cost factors will be considered below. Figure 7.5 shows the ASG in which each arc is associated with the cost of the assembly task.

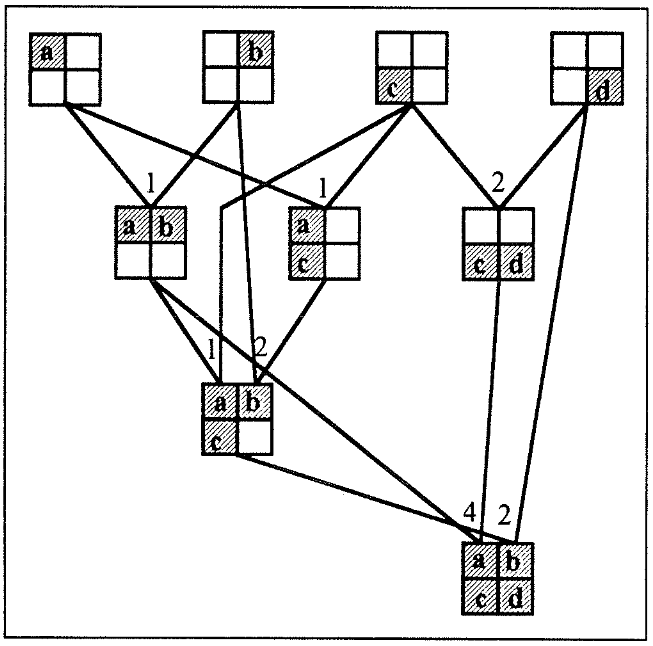


Figure 7.5. An ASG associated with cost factors

The ASG in Figure 7.5 encompasses three different trees: 1. {(a)-(b)-(c)-(d)}, 2. {(a)-(c)-(b)-(d)}, 3. {(a-b)-(c-d)}. The first two are linear sequences and the third one is with subassemblies. The total assembly cost is the sum of all the costs in that tree. The total cost for each of these sequences are 4, 5 and 7 units respectively. The first sequence is the one with the least and will be considered as the best choice in this case.

For automatic assembly, where assembly of part i to part j is different from the assembly of part j to part i, the assembly costs will be assigned to individual arcs instead of assigning to the hyperarcs. That means, each hyperarc is associated with two different weight factors. This can be clearly distinguished by assigning two different cost factors to the line joining the nodes {a} and {a,b} and to the line joining the nodes {b} and {a,b} in Figure 7.5.

The selection and evaluation criteria discussed above covers most of the real life industrial situations. The best plan can be chosen by enumerating all the sequences using the chosen criteria. The editing process of ASG and AST together provide an efficient method of pruning the large number of available sequence choices. This simplifies the job of the assembly planner to come up with the most efficient and practicable assembly sequence to assemble the product.

CHAPTER 8

COMPUTER INTEGRATED ASSEMBLY PLANNING SYSTEM (CIAPS)

8.1 INTRODUCTION

Based on the procedures and algorithms described in the previous chapters, a Computer Integrated Assembly Planning System (CIAPS) is proposed. The emphasis of the CIAPS is on generating assembly plans automatically from the assembly models. The input for this system is a geometric model of the assembly and the output is an optimal assembly sequence, which will specify the order and the direction of the assembly tasks. Because of the complexity and combinatorial explosive nature of the problem, the following assumptions are made in the design of this system.

- All component parts are rigid bodies.
- One component or subassembly is added at one time.
- No change in shape of the part as a result of the assembly operation (for example, springs, C-rings, etc., are not considered)
- All components are assembled directly to their final positions with a single linear motion. Actions such as "insert and twist" are not allowed.
- Stability constraints are satisfied. The components stay in their assembled positions and remain fixed during subsequent operations.
- Parts can be assembled in only six possible directions (i.e., parallel to the cartesian coordinate axes).

These assumptions reduce the complexity of the problem. Nevertheless, they are practical enough to suit many real-life situations.

8.2 ARCHITECTURE OF CIAPS

Figure 8.1 illustrates the architecture of CIAPS from its top-level in a modular perspective. CIAPS is divided into five basic modules. The division of these modules and their functions are described in Table 8.1.

Table 8.1. Modules of the CIAPS and their functions

Name of the module	Function of the module
DESIGN	Creation of solid models of assembly
EXTRACT	Extraction of precedence knowledge in terms of C and T functions
GENERATE	Generation of all possible sequences in ASG representation
SELECT	Pruning the ASG to select a few candidate sequences
EVALUATE	Economic analysis

To meet the specified needs, first, the solid models of the assembly are created. The system then analyses the geometry of components, discovers mating conditions, obtains obstruction directions of components, discovers precedence relationships, generates assembly sequences, selects the candidate sequences and determines the optimal sequence. All these modules of the system and how they perform the specified functions will be discussed in detail in the following sections.

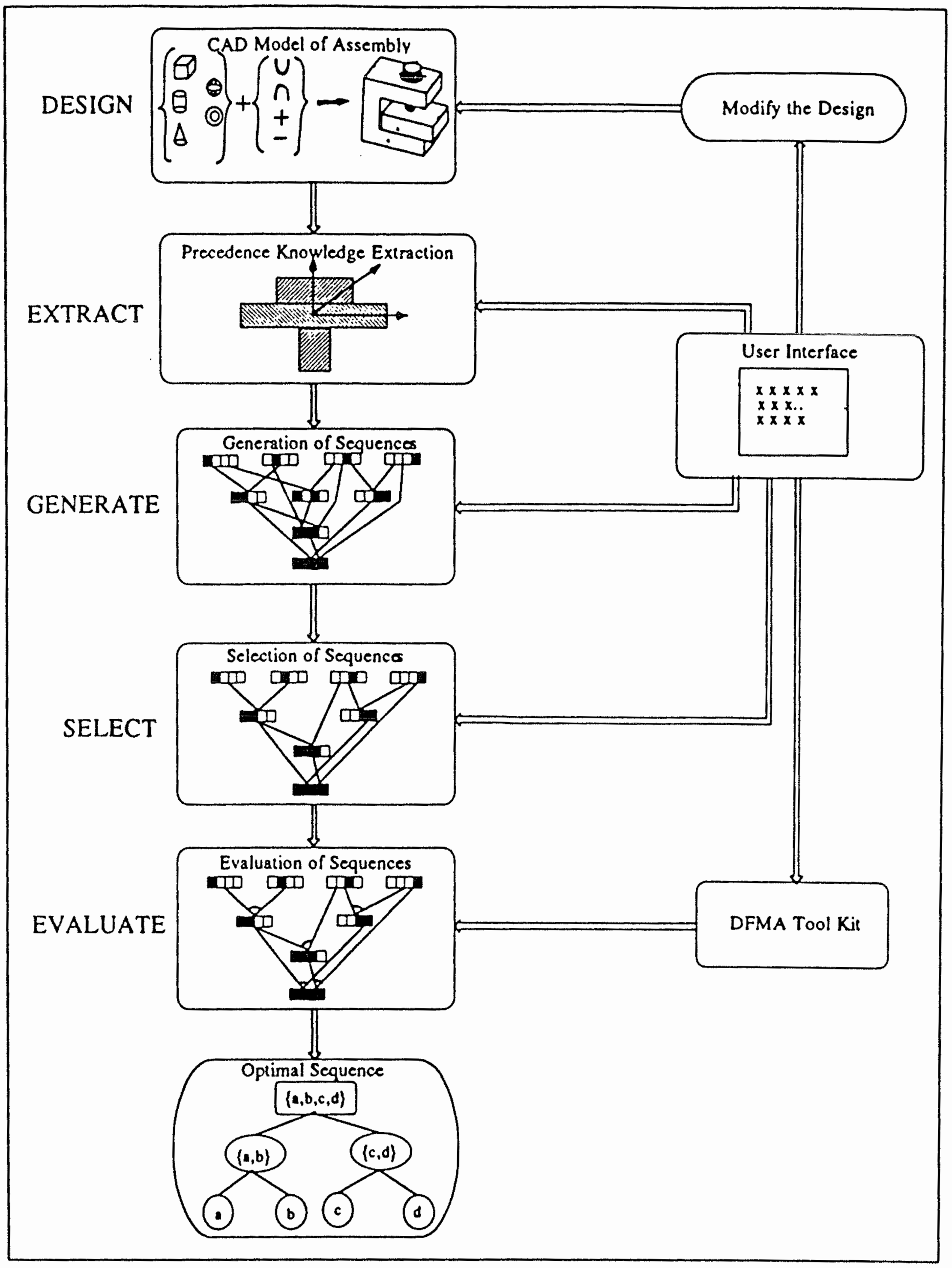


Figure 8.1. Architecture of CIAPS

8.2.1 Module DESIGN

The first phase of CIAPS is the creation of geometric model of the assembly that describes the component parts and the spatial relationships among them. The procedure is to create the solid models for individual components of assembly based on the global coordinate system of PADL-2. The parts are actually described in their respective locations in the assembled position. Solid models of all the parts are created by specifying the size and position parameters of part primitives with respect to the global coordinate system. The final geometric assembly model is a collection of component solids in their designated positions.

The DESIGN module allows the user to create the solid models of the components of assembly using the functions provided by PADL-2. The models of these individual components are then combined to create a model of the assembly.

The user provides PADL-2 definition of the assembly and then stores it into a file called the input file. The first line of the input file includes the number of statements used to create the assembly and the number of parts in the assembly. The program reads these statements and then converts them into PADL-2 code by calling the PADL-2 routine 'PP2'. Then it calls the display routine to display the assembly. The display screen is set up with four windows to display the top view, the front view, the side view and the PADL-2 default isometric view. The flow-chart for the DESIGN module is illustrated in Figure 8.2. A computer program was written in FORTRAN to communicate with PADL-2, to read PADL definition of the assembly, to translate the assembly definition into PADL-2 code and to display the assembly.

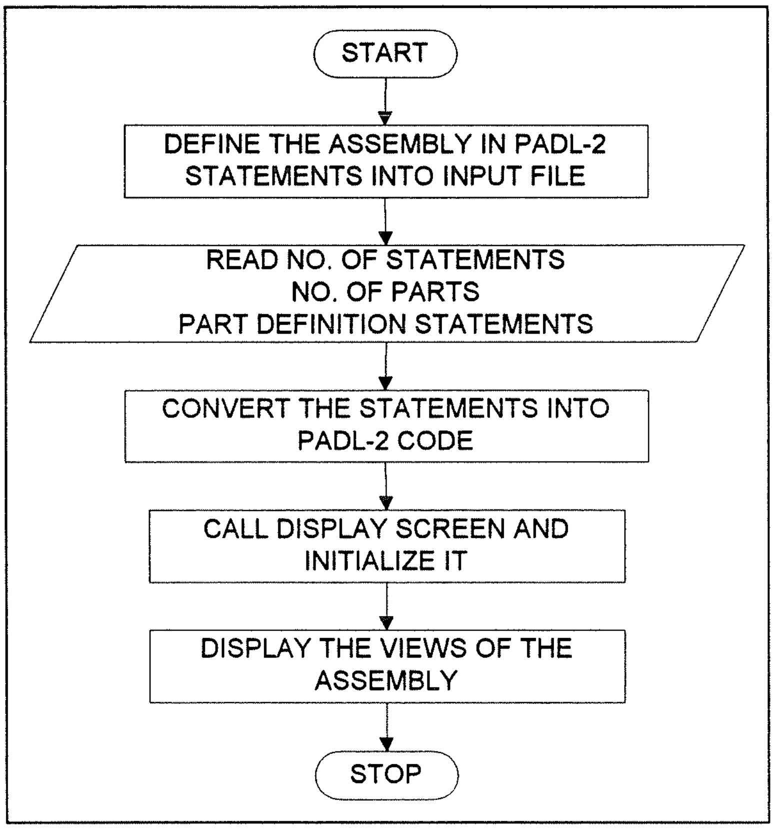

START
DEFINE THE ASSEMBLY IN PADL-2 STATEMENTS INTO INPUT FILE
READ NO. OF STATEMENTS NO. OF PARTS PART DEFINITION STATEMENTS
CONVERT THE STATEMENTS INTO PADL-2 CODE
CALL DISPLAY SCREEN AND INITIALIZE IT
DISPLAY THE VIEWS OF THE ASSEMBLY
STOP

Figure 8.2. Flowchart for the DESIGN module

8.2.2 Module EXTRACT

The purpose of this module is to obtain the precedence relationships among the components. This module interacts with the design of the assembly created on the PADL-2 solid modeling system and employs the collision detection algorithm for every component with respect to the other. Basically it uses the interference detection facility and mass property calculator of PADL-2. The program executes a translation of each component along the six main directions in order to recognize the interference and thus the direction of disassembly. Figure 8.3 illustrates the collision detection algorithm to recognize the contacts between two components.

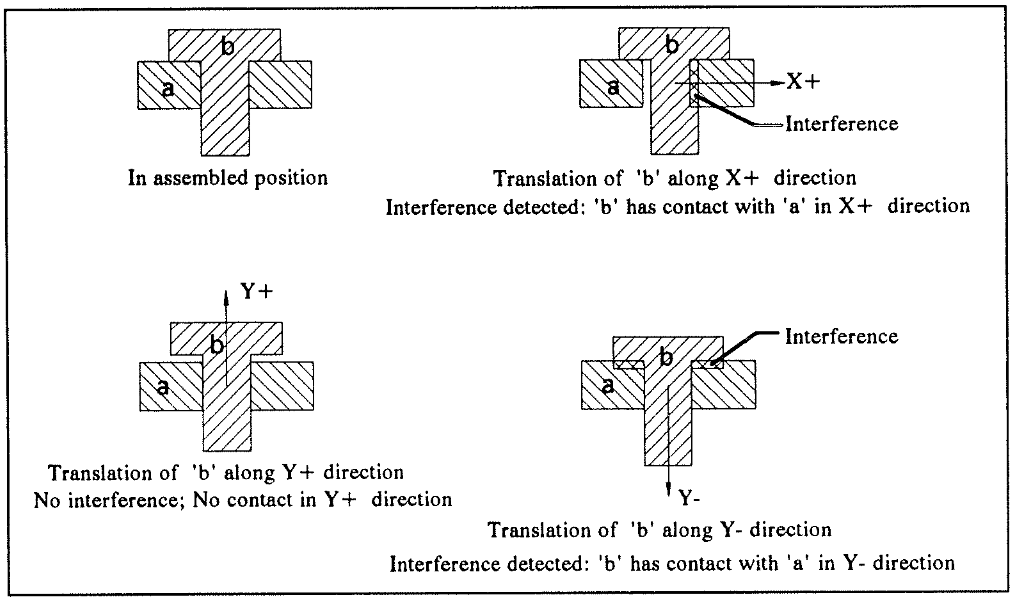

b
a
In assembled position
X+
Interference
Translation of 'b' along X+ direction
Interference detected: 'b' has contact with 'a' in X+ direction
Y+
Translation of 'b' along Y+ direction
No interference; No contact in Y+ direction
Interference
Y-
Translation of 'b' along Y- direction
Interference detected: 'b' has contact with 'a' in Y- direction

Figure 8.3. Recognition of contacts by PADL-2

The steps and the logic used in the extraction of C and T functions is illustrated in the flowcharts shown in Figure 8.4 and Figure 8.5 respectively. As PADL-2 has facilities to interact with the FORTRAN routines, the code for the EXTRACT module was written in FORTRAN. The listing of the computer code for the DESIGN and the EXTRACT modules is presented in the Appendix.

It can be observed that the logic used in the extraction of both the C and T functions is the same. In both cases, one part is moved over a certain distance while keeping the other fixed and tested for the presence of an intersection. In the case of the C-function, the increment is given only once, whereas for the T-function, the increment is given more than once until it reaches the specified maximum limit. This maximum limit can be taken as the dimension of a parallelepiped that encompasses the whole assembly. The output of the EXTRACT module is the relational information for each pair of components.

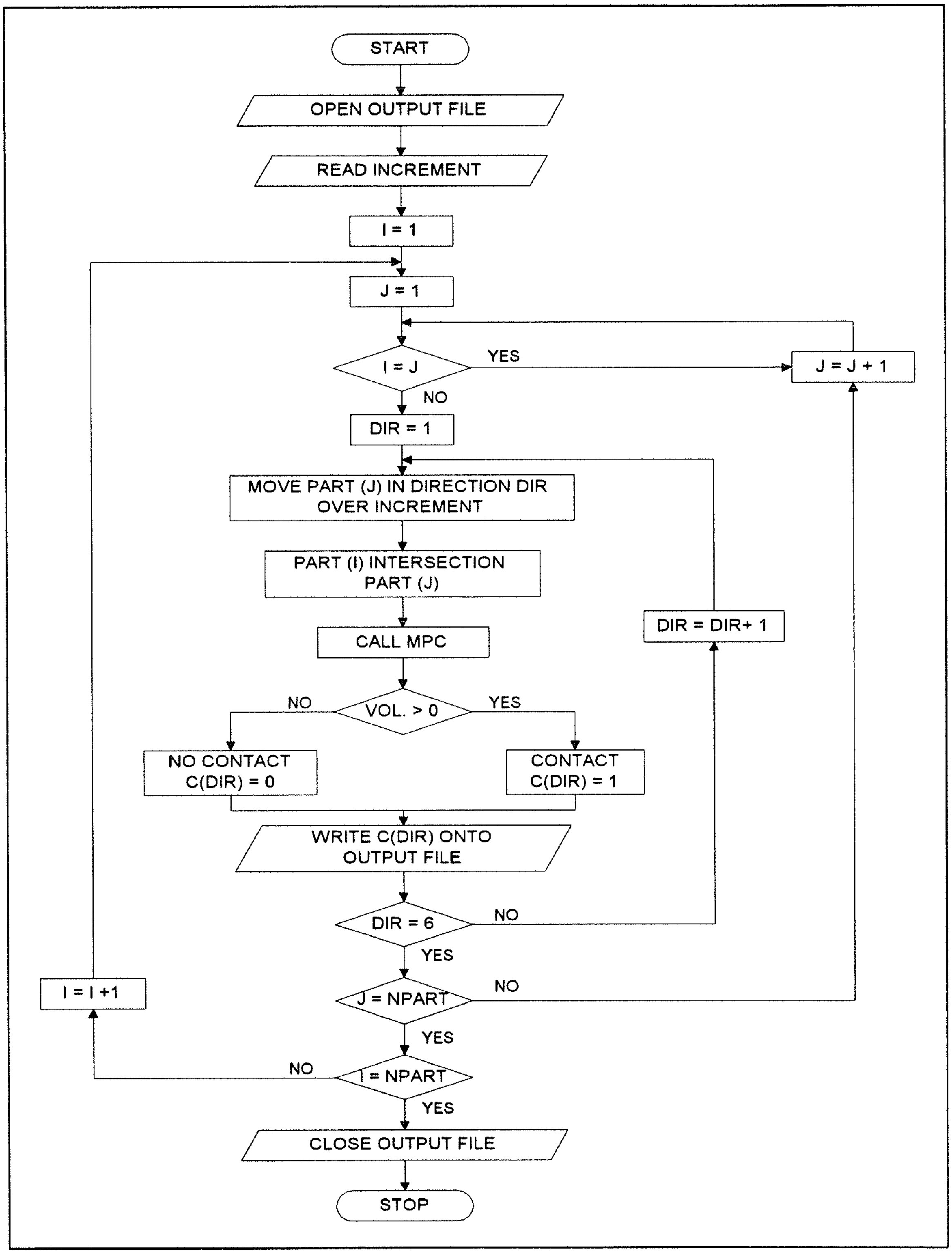


Figure 8.4. Flowchart for the extraction of the C-function

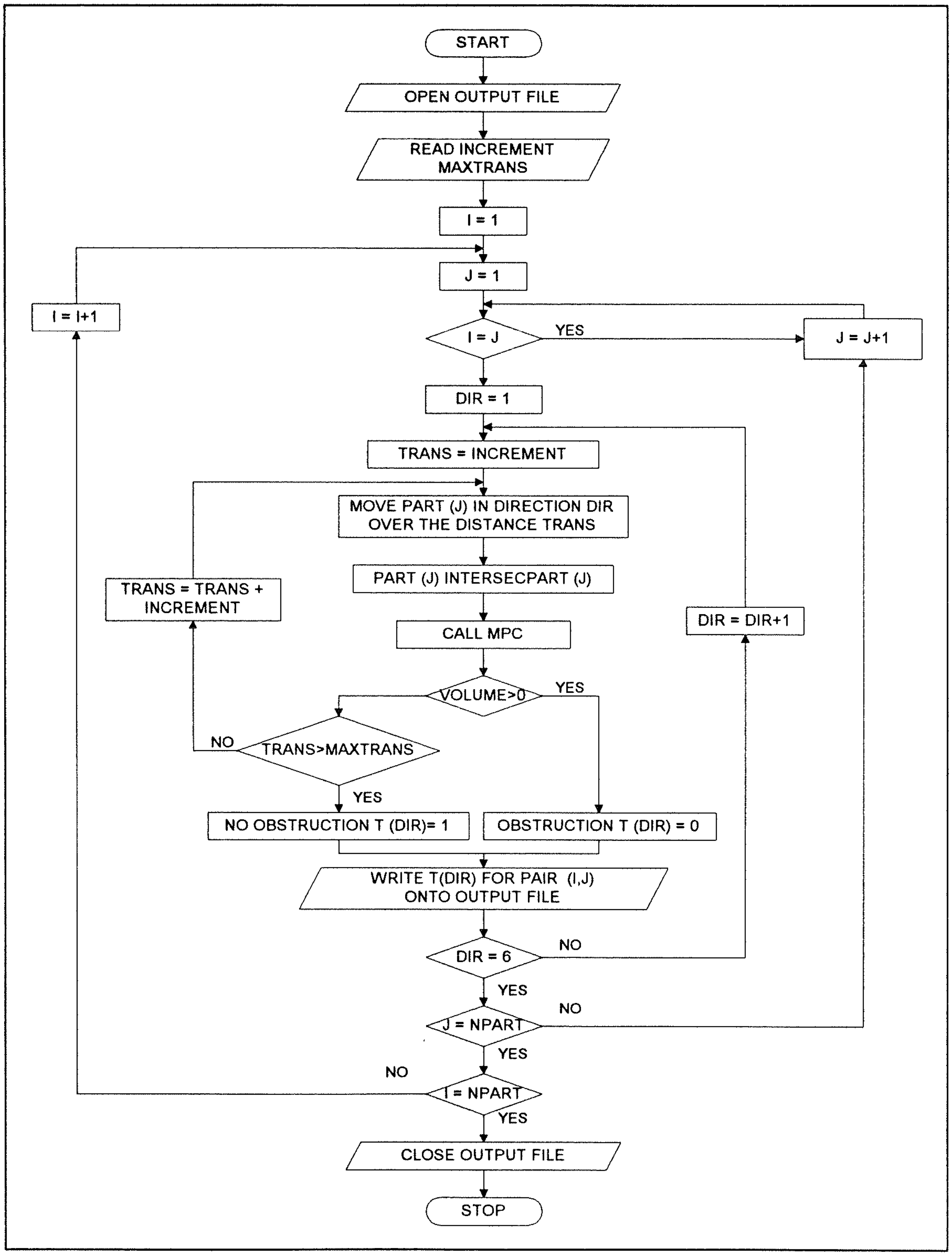


Figure 8.5. Flowchart for the extraction of the T-function

It must be emphasized how this method can yield a wrong information if the clearance existing among the components is greater than the translation increment used during recognition of contacts. Another drawback of using small increments is that it increases the number of increments necessary to check the interference in order to find the collision information. Consequently, it increases the computational time considerably. Nevertheless this increment cannot be too great, so that the elements with a smaller thickness are not bypassed by the translating element. Figure 8.6 clearly depicts the effect of the value of this increment. If the translation of component a in the X+ direction is less than the clearance (0.01), it will indicate that there is no contact between the components in the X+ direction. On the other hand, if the increment for the translation of b in the direction Y- is greater than 2.5 (thickness of component a), it will say that there is no contact. So, in this case, the increment must be between 0.01 and 2.5.

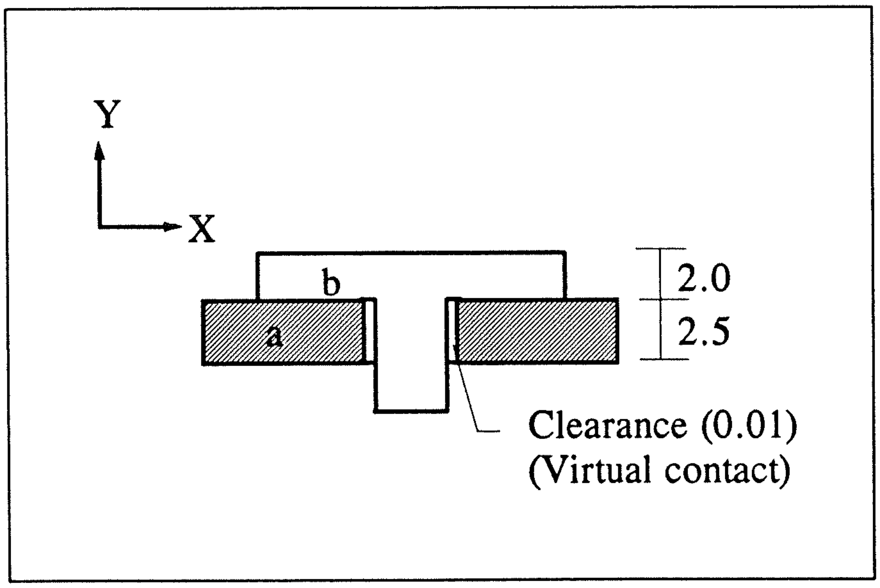
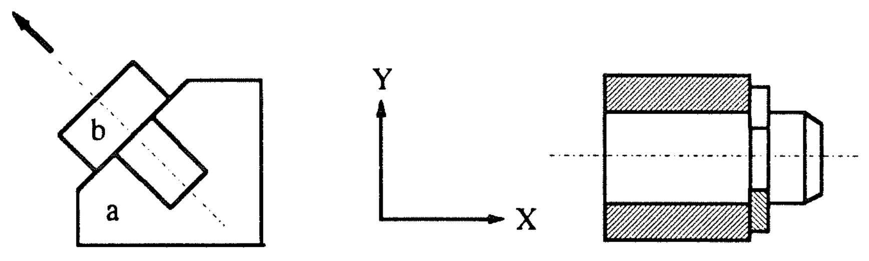


Figure 8.6. Effect of incremental distance

For these reasons, the choice of the increment parameter is critical, and it must result in:

maximum clearance < increment < minimum thickness.

Therefore, this method is correctly applicable when the thickness of the components is greater than the maximum clearance of the connections. The default value of this increment has been fixed at 0.05 mm, which is appropriate in many cases. However, this value should be adjusted according to the features of the product to be assembled.



(a) An assembly whose assembly direction is not oriented along one of the principal axes

(b) An example of a C-ring which does not have any specific disassembly direction

Figure 8.7. Example assemblies violating the assumptions of CIAPS

During the execution of the module, in some cases, it may happen that the program does not find the disassembly direction of a component, when the component has a disassembly direction different from the main directions as depicted in Figure 8.7a. This could result in erroneous precedence relationships. This problem can be eliminated with some human intervention or by defining more number of coordinate axes along which the components can be separated from others. But this seriously increases the complexity as it requires that collision detection be performed along all directions. However, this situation should not be frequent if the "Design For Assembly" criteria are applied during the design phase.

This module will make wrong inferences, if there is any deformation or change in the shape or size of the components during assembly. For example, springs get compressed or stretched during assembly. Since the models of the components are created in their assembled state, it does not pose any serious problem. Figure 8.7b shows an example of a C-ring, where the shape of the component after assembly is different from its original shape. In this case, interactive human intervention would be needed to obtain the correct result.

It is also observed that the maximum translation distance to extract the T-functions and the PROP-LEVEL (78) have considerable effect on the program execution time. Here the PROP-LEVEL specifies the level of subdivision. As the level of subdivision is increased, the number of cells used to form the object increases and the minimum size of the cell decreases. Usually, for an increase of 1 in the PROP-LEVEL, the execution time increases by four to five times.

Another drawback with the PADL-2 is that it keeps all the calculations in its memory as a result of which the memory gets exhausted. Especially, while extracting the T-functions, the program does several iterations for each pair. After doing some iterations, the execution of the program stops because of insufficient memory. To overcome this problem, for assemblies with a large number of parts, the program must be run several times, each time for a set of pairs. The results will be stored in the output files and all these files must be concatenated into one file to get the final result.

Another observation is that PADL-2 gets stuck while doing intersection operation between components having a common surface. Hence, it is preferable to maintain a minimum clearance between the components while defining the assembly. Also, for the components defined by the primitives in the same volume of space, the determination of intersection takes longer time.

The practical application of this method has some limitations. If the execution is entirely automatic, the time required could be quite long, since for each component, the absence of collisions must be detected along several directions to find the proper path in order to assemble it to the other components. If the disassembly direction of each component is given interactively by the user, the system is constrained to follow certain solutions, whereas others are neglected. But, the level of automation is reduced in that case. The method implemented here can be considered as a compromise between the fully automated and the interactive approaches. A substantial part of the execution is automated, but some interactive inputs are required where the decision process of the human mind is faster and more efficient. Since the system is not comletely integrated, interactive means are not implemented at present time.

8.2.3 Module GENERATE

This module generates all the possible sequences by which each component can be assembled to obtain the final product and represents them in the form of an AST. This module directly uses the precedence knowledge base generated from the module EXTRACT, containing the information about the contacts and mobility among the components. The proposed procedure also provides the assembly direction to obtain all possible ways of assembly of the component. The system generates only the feasible sequences by deleting those which do not satisfy the assembly precedences. This procedure can be made more efficient by linking the evaluation of stability test for each subassembly. The algorithm used to develop this module has been discussed in Chapter 5 and the algorithm for the ASG construction has been explained in Chapter 6. The computer code for this module has been written in PASCAL.

Figure 8.8 describes all the steps involved in the generation of sequences from the relational knowledge supplied by the previous module. It takes all the parts, the pairs, and the C and T functions as input. Then it creates N nodes at level 1 representing the individual components of the assembly. In the next step, the system lists all the pairs having contact at least in one direction. Actually there is no need to check for contact for all the M pairs, because among the M ordered pairs supplied to the module, the last M/2 pairs are just duplications of the first M/2 pairs, if the order is not taken into account. As the contact checking is performed for M/2 pairs only, there is a considerable reduction in the required computational time.

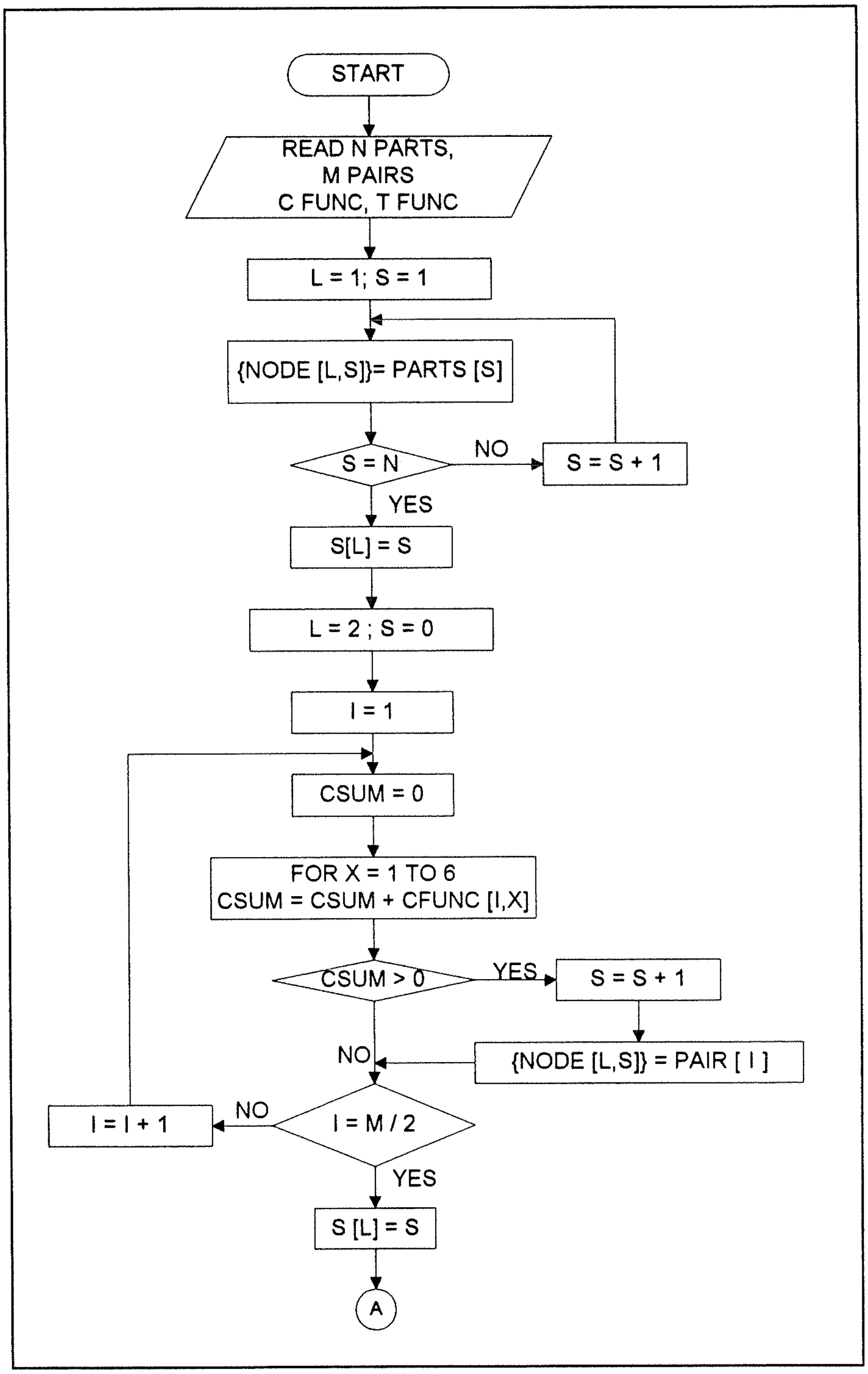


Figure 8.8. Flowchart for the GENERATE module

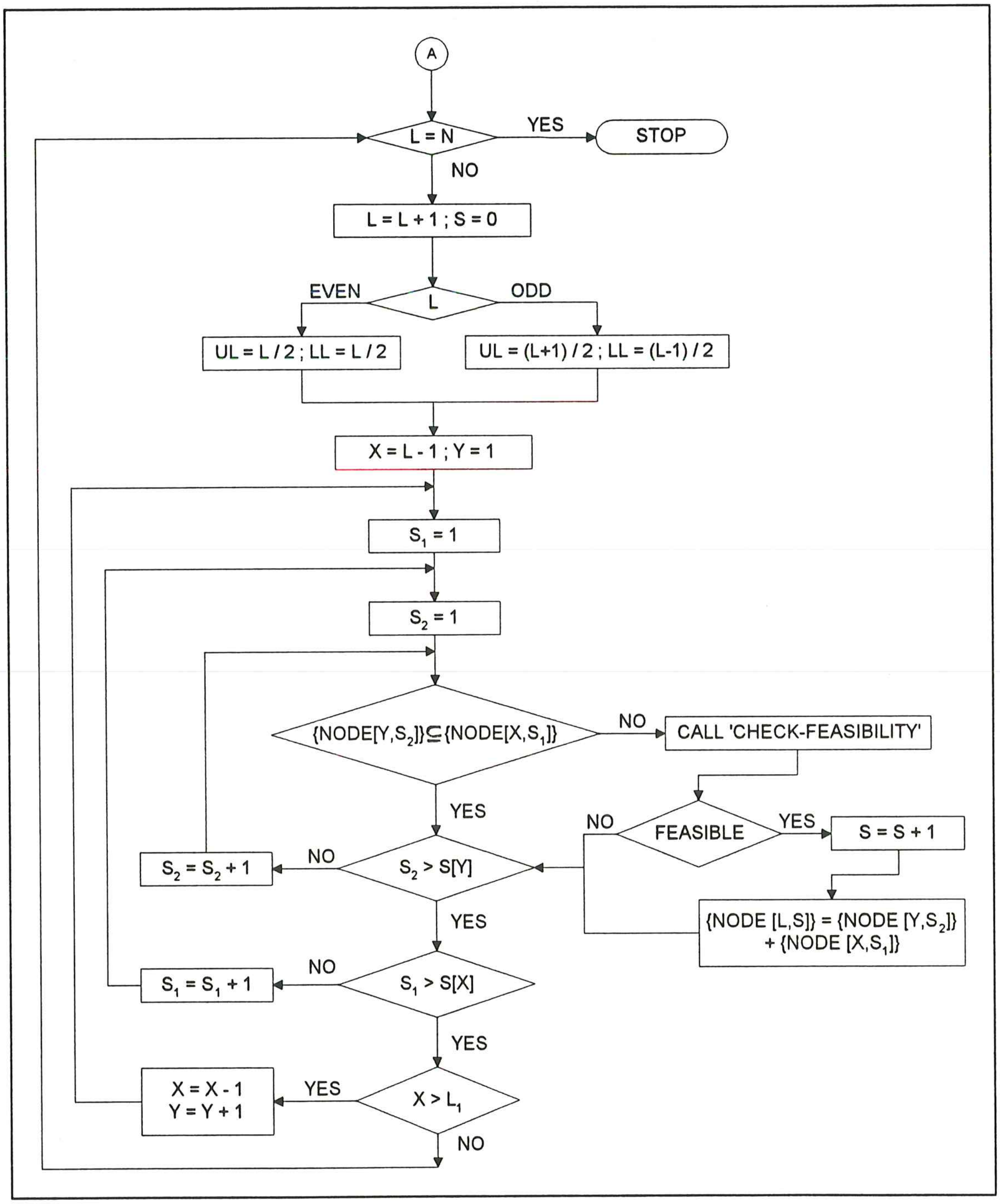


Figure 8.8. (Continuation)

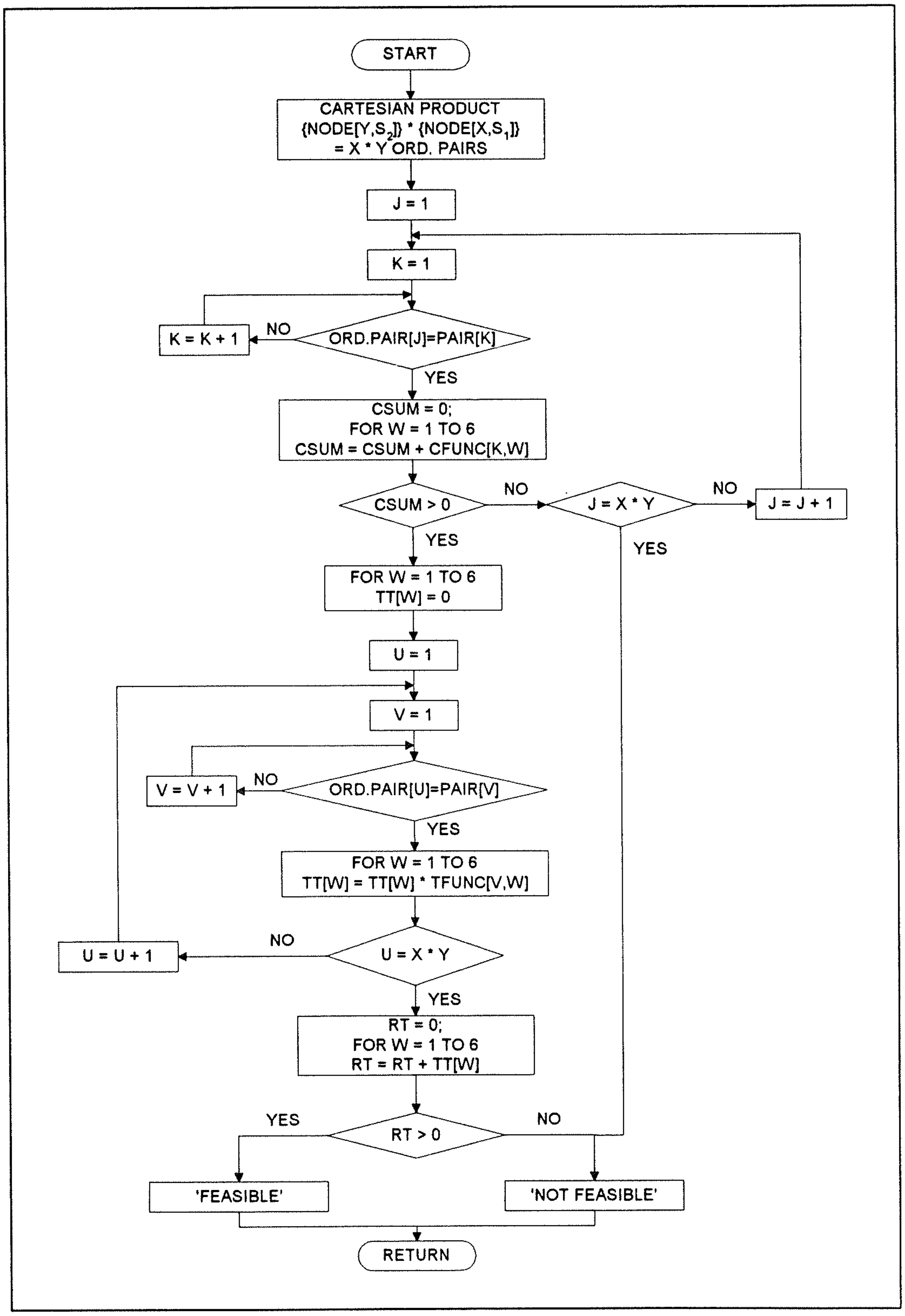


Figure 8.9. Flowchart for the subroutine "Check-feasibility"

In the next step, this module generates the higher order subassemblies in the subsequent levels that can be formed from the previously generated subassemblies and individual components. Here, to be feasible, an assembly must satisfy both the connectivity and precedence constraints. The flowchart for the subroutine "check-feasibility" which is shown in Figure 8.9, explains how the feasibility of the assembly is verified. This subroutine takes two subassemblies to be assembled and analyzes whether the assembly of the two subassemblies is feasible or not, as discussed in the earlier chapters. Once the module reaches the last level, i.e., the final assembly, the execution stops. All the generated assembly states and tasks are then tabulated into AST. The corresponding ASG is drawn using AUTOCAD software.

8.2.4 Module SELECT

The generated sequences can be large in number, slowing down the further computational process of selection of the best assembly plan. For this reason, it is imperative to reduce this number, deleting all the unstable assembly states and infeasible assembly tasks. The criteria described in Chapter 7 can be used interactively to reduce the sequence count and to select a few candidate sequences for further analysis. The assembly planner (user) will identify the selection criteria based on his experience and available facilities. Then these criteria will be applied to reduce the number of sequences. When an unwanted assembly state or unwanted assembly task is identified, the corresponding box or arc is deleted from the ASG (or AST). Then the graph is updated by deleting resulting dead nodes, if there are any. The procedure which is explained in Chapter 7 for the reduction of the sequence count, is illustrated in Figure 8.10. The editing facilities of AUTOCAD are used to edit the ASG.

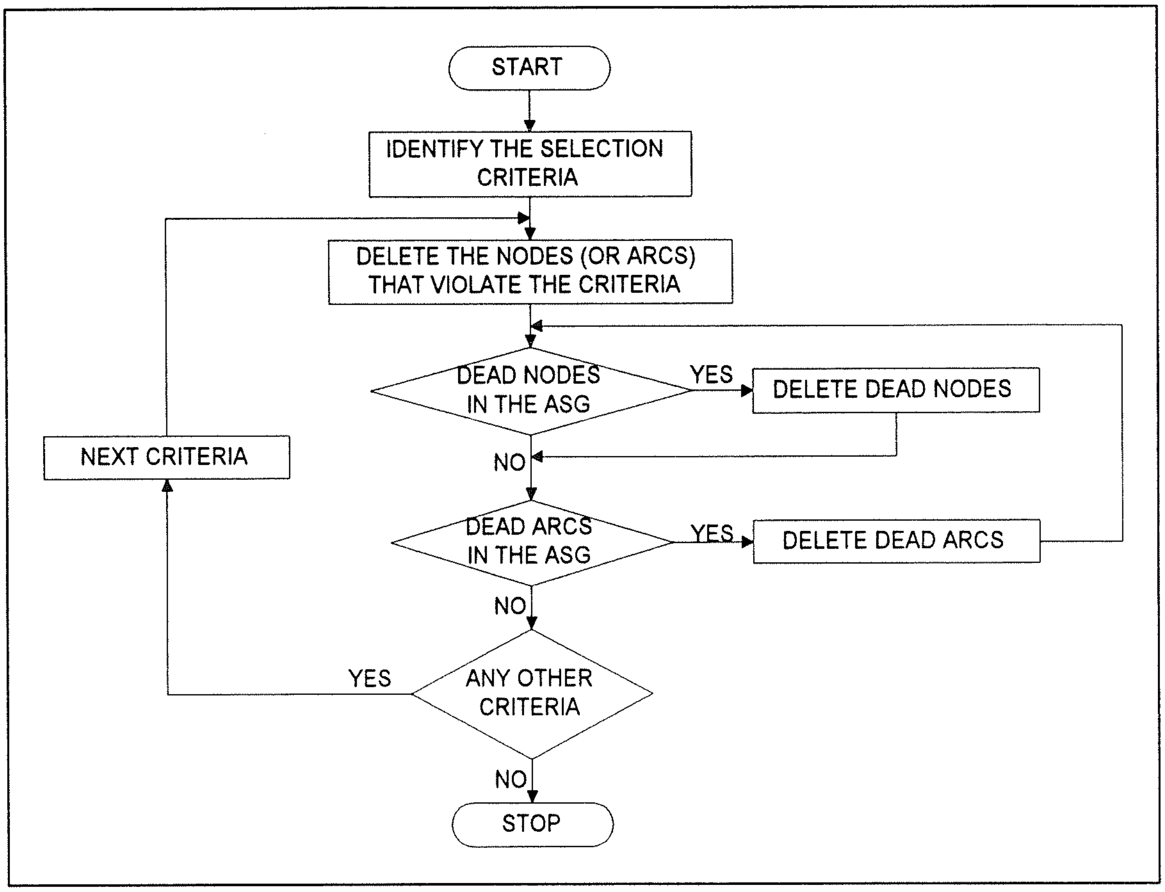


Figure 8.10. Flowchart for the SELECT module

8.2.5 Module EVALUATE

This module uses the DFA software to estimate the assembly time and assembly cost based on the degree of difficulty of the assembly. The assembly time and cost of each assembly task is estimated using the DFA Toolkit software and are assigned to the corresponding arcs in the ASG. Then the sum of the costs in each path in the ASG will be calculated, which gives the total assembly cost, if the product is assembled using that sequence. The path with minimum assembly cost will be taken as the final choice. The basic steps involved in this module are shown in Figure 8.11. In essence, the output of this module is the best assembly sequence.

The DFA evaluation technique has been proven to be a systematic approach to evaluate a design from the assembly point of view. The DFA Toolkit performs all the data manipulation and evaluation. The user has to determine the part assembly features such as size, end-to-end (alpha) symmetry, axial (beta) symmetry, requirement of extra tooling, etc. The size and thickness of the part can be found by finding the smallest rectangular (or cylinder) envelope surrounding the part. The size of the part is equal to the largest dimension of the envelope, while the smallest dimension of the envelope gives the thickness of the part. Li and Hwang (42) have attempted to automate the entire DFA evaluation procedure by determining the part assembly features automatically from the design of the product.

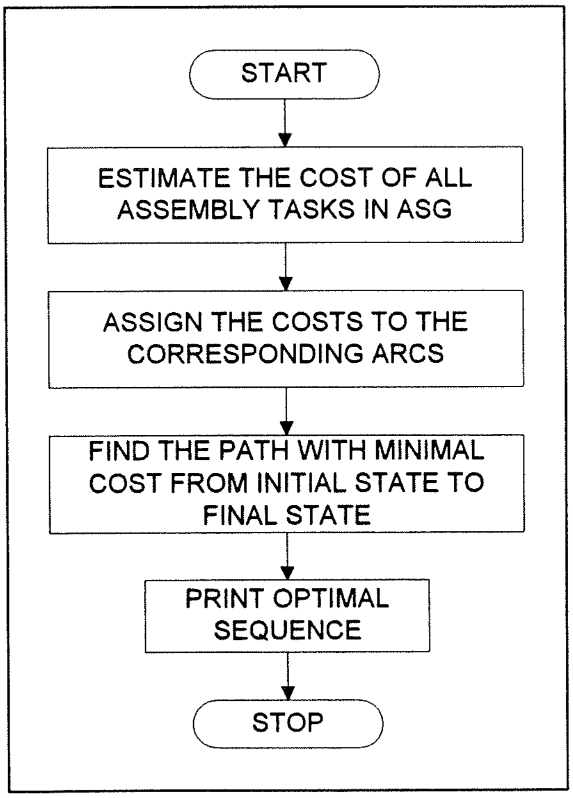

START
ESTIMATE THE COST OF ALL ASSEMBLY TASKS IN ASG
ASSIGN THE COSTS TO THE CORRESPONDING ARCS
FIND THE PATH WITH MINIMAL COST FROM INITIAL STATE TO FINAL STATE
PRINT OPTIMAL SEQUENCE
STOP

Figure 8.11. Flowchart for the EVALUATE module

The system also has a user interface, so that the user can interact at any time to modify the design, to impose constraints or to take any necessary decision. The design of the product can be modified according to the Design for assembly principles described by Boothroyd and Dewhurst (6).

The choice of how much detail to include in the plan is a critical one in the design of the assembly planning system. More detailed plans can be evaluated by more accurate and realistic cost criteria, so better decisions can be made. But more detailed plans also require more decisions to be made, so combinatorial explosion may limit the degree to which the plans can be optimized. Finding the right trade-off will be a fundamental issue in the assembly planning research.

The approach proposed in the design of CIAPS enables us to have a global view of the entire assembly planning system. It exhibits superiority over existing sequence generation methods in three areas : (1) It allows a designer to investigate the assembly characteristics of the product right in the early development phase and hence it is useful for concurrent engineering environment; (2) It permits a multitude of component assembly directions; (3) It incorporates a part collision detection algorithm to extract precedence constraints. Thus, in addition to the generation of feasible assembly plans, this method provides the detailed process information concerning the base part selected for the assembly task, mating directions available, etc. The information about mating directions and mating parts can be used to align a part for the assembly task.

CHAPTER 9

APPLICATIONS

9.1 INTRODUCTION

This chapter will present examples to illustrate the applicability and validity of the sequence determination methodology discussed in the previous chapters. To demonstrate the usefulness of the proposed approach in solving the problems that are of interest to industry, and also to compare the present approach with the previously developed methods, the following two examples are considered.

1. Ball-point pen assembly
2. Built-in oven door assembly

9.2 BALL-POINT PEN ASSEMBLY

The ball-point pen example was first used by Bourjault (6) and later several researchers used it to explain their sequence determination methods. Hence, this product is considered as a good example to compare the present method with others. Figure 2.2 presents the components of a typical "use and throw" plastic ball-point pen assembly. Basically, it consists of six components: Body (b), Cap (c), Button (u), Head (h), Tube (t) and Ink (i). For the sake of convenience, each component type is given a unique code, which are shown in the parentheses. These code letters will be used to denote the components in the sequence generation and representation.

The geometric models of individual components are created in their assembled position in 3-D space. The DESIGN module displays all four views of the pen as shown in the Figure 9.1. It should be noted that, although ink is liquid, it is assumed as a solid component, according to the assumptions used in the design of CIAPS. Details of geometry and dimensions include that the tube has a clearance-fit with the body and that the head has a clearance fit with the cap, so that, neither of the pairs of these parts form mated pairs.

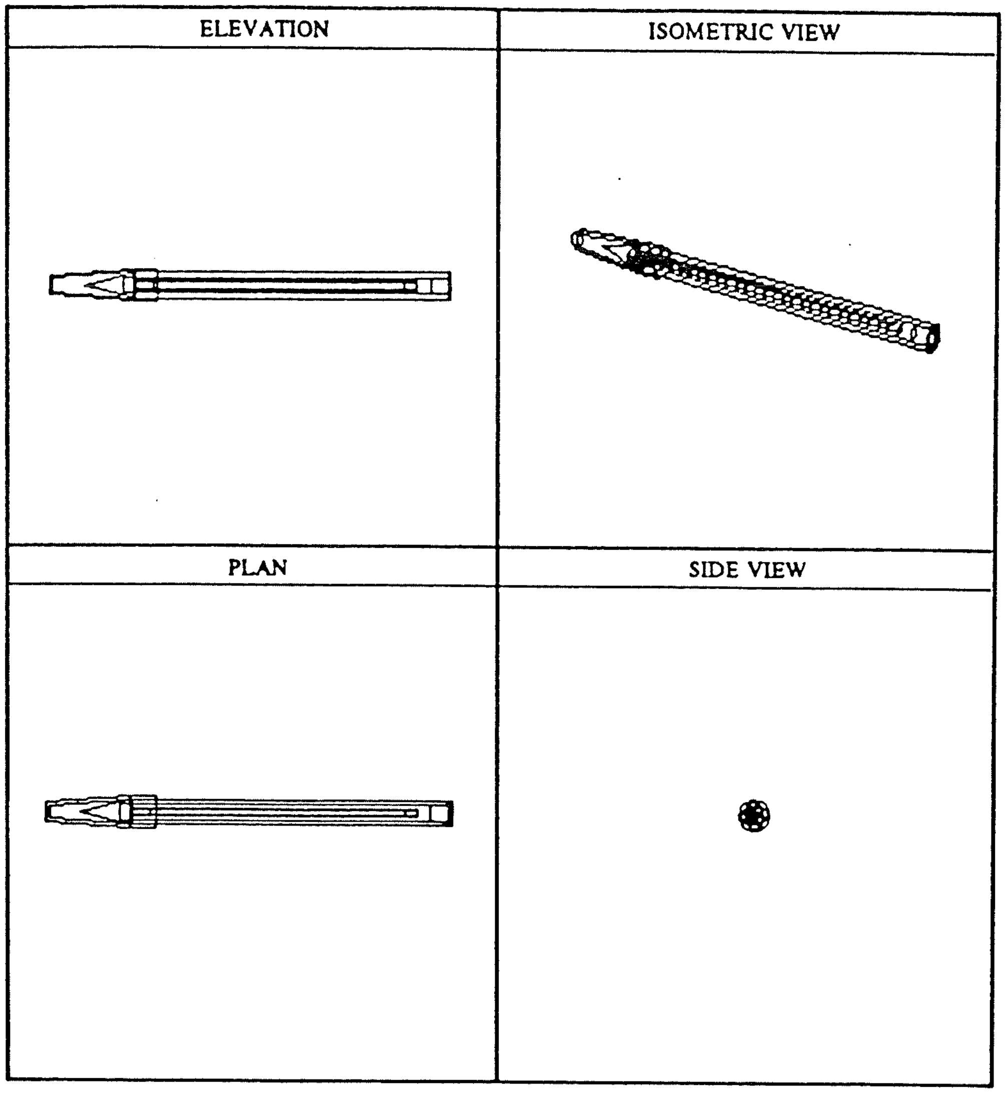


Figure 9.1. PADL-2 views of the ball-point pen assembly

9.2.1 Precedence Knowledge Extraction

Once the geometric model of the assembly is created, the next step is the extraction of the C and T functions. After applying the collision detection procedure, the resulting C and T functions (the output of the EXTRACT module) for all the pairs of components are presented in Table 9.1. Since there are six components in the pen assembly, there are total of 15 pairs in the left half of the table. The other 15 pairs in the right half of the table are just inverse pairs and they are same as the first 15 pairs when order is not considered.

Table 9.1. C & T functions for the Ball-point pen assembly

Pair	C-Function						T-Function						Pair	C-Function						T-Function					
	C_1	C_2	C_3	C_4	C_5	C_6	T_1	T_2	T_3	T_4	T_5	T_6		C_1	C_2	C_3	C_4	C_5	C_6	T_1	T_2	T_3	T_4	T_5	T_6
(c,b)	0	1	1	1	1	1	1	0	0	0	0	0	(b,c)	1	1	1	0	1	1	0	0	0	1	0	0
(c,u)	0	0	0	0	0	0	1	1	1	0	1	1	(u,c)	0	0	0	0	0	0	0	1	1	1	1	1
(c,h)	0	0	0	0	0	0	1	0	0	0	0	0	(h,c)	0	0	0	0	0	0	0	0	0	1	0	0
(c,t)	0	0	0	0	0	0	1	0	0	0	0	0	(t,c)	0	0	0	0	0	0	0	0	0	1	0	0
(c,i)	0	0	0	0	0	0	1	0	0	0	0	0	(i,c)	0	0	0	0	0	0	0	0	0	1	0	0
(b,u)	0	1	1	1	1	1	1	0	0	0	0	0	(u,b)	1	1	1	0	1	1	0	0	0	1	0	0
(b,h)	1	1	1	0	1	1	0	0	0	1	0	0	(h,b)	0	1	1	1	1	1	1	0	0	0	0	0
(b,t)	0	0	0	0	0	0	1	0	0	1	0	0	(t,b)	0	0	0	0	0	0	1	0	0	1	0	0
(b,i)	0	0	0	0	0	0	1	0	0	1	0	0	(i,b)	0	0	0	0	0	0	1	0	0	1	0	0
(u,h)	0	0	0	0	0	0	0	1	1	1	1	1	(h,u)	0	0	0	0	0	0	1	1	1	0	1	1
(u,t)	0	0	0	0	0	0	0	1	1	1	1	1	(t,u)	0	0	0	0	0	0	1	1	1	0	1	1
(u,i)	0	0	0	0	0	0	0	1	1	1	1	1	(i,u)	0	0	0	0	0	0	1	1	1	0	1	1
(h,t)	0	1	1	1	1	1	1	0	0	0	0	0	(t,h)	1	1	1	0	1	1	0	0	0	1	0	0
(h,i)	0	1	1	1	1	1	1	0	0	0	0	0	(i,h)	1	1	1	0	1	1	0	0	0	1	0	0
(t,i)	0	1	1	0	1	1	1	0	0	1	0	0	(i,t)	0	1	1	0	1	1	1	0	0	1	0	0

9.2.2 Sequence Generation

In this step, the sequence generation algorithm (GENERATE module) takes the contact and precedence information from the above table as input and generates all the geometrically feasible assembly states and tasks. Table 9.2 presents these assembly states and the corresponding state transitions. These assembly states and tasks are then mapped on to the assembly sequence graph as shown in Figure 9.2.

Table 9.2. The AST for the ball-point pen assembly

Level	Nodes (States)	Arcs (Tasks)
2	{h,t}	{(h), (t)}
	{t,i}	{(t), (i)}
	{h,i}	{(h), (i)}
	{h,b}	{(h), (b)}
	{b,u}	{(b), (u)}
3	{h,t,b}	{(h,t), (b)}; {(h,b), (t)}
	{h,t,i}	{(h,t), (i)}; {(h,i), (t)}; {(t,i), (h)}
	{h,i,b}	{(h,i), (b)}; {(h,b), (i)};
	{h,b,c}	{(h,b), (c)}
4	{h,t,b,c}	{(h,t,b), (c}; {(h,b,c), (t)}
	{h,t,i,b}	{(h,t,b), (i)}; {(h,t,i), (b)}; {(h,i,b), (t)} {h,b), (t,i)}
	{h,i,b,c}	{(h,i,b), (c)}; {(h,b,c), (i)}
5	{h,t,i,b,c}	{(h,t,b,c), (i)}; {(h,t,i,b), (c)}; {(h,i,b,c), (t)} {(h,b,c), (t,i)}
	{h,t,i,b,u}	{(h,t,i,b), (u)}; {(h,t,i), (b,u)}
6	{h,t,i,b,u,c}	{(h,t,i,b,c), (u)}; {(h,t,i,b,u), (c)}

Total number of states = 15 Total number of tasks = 29

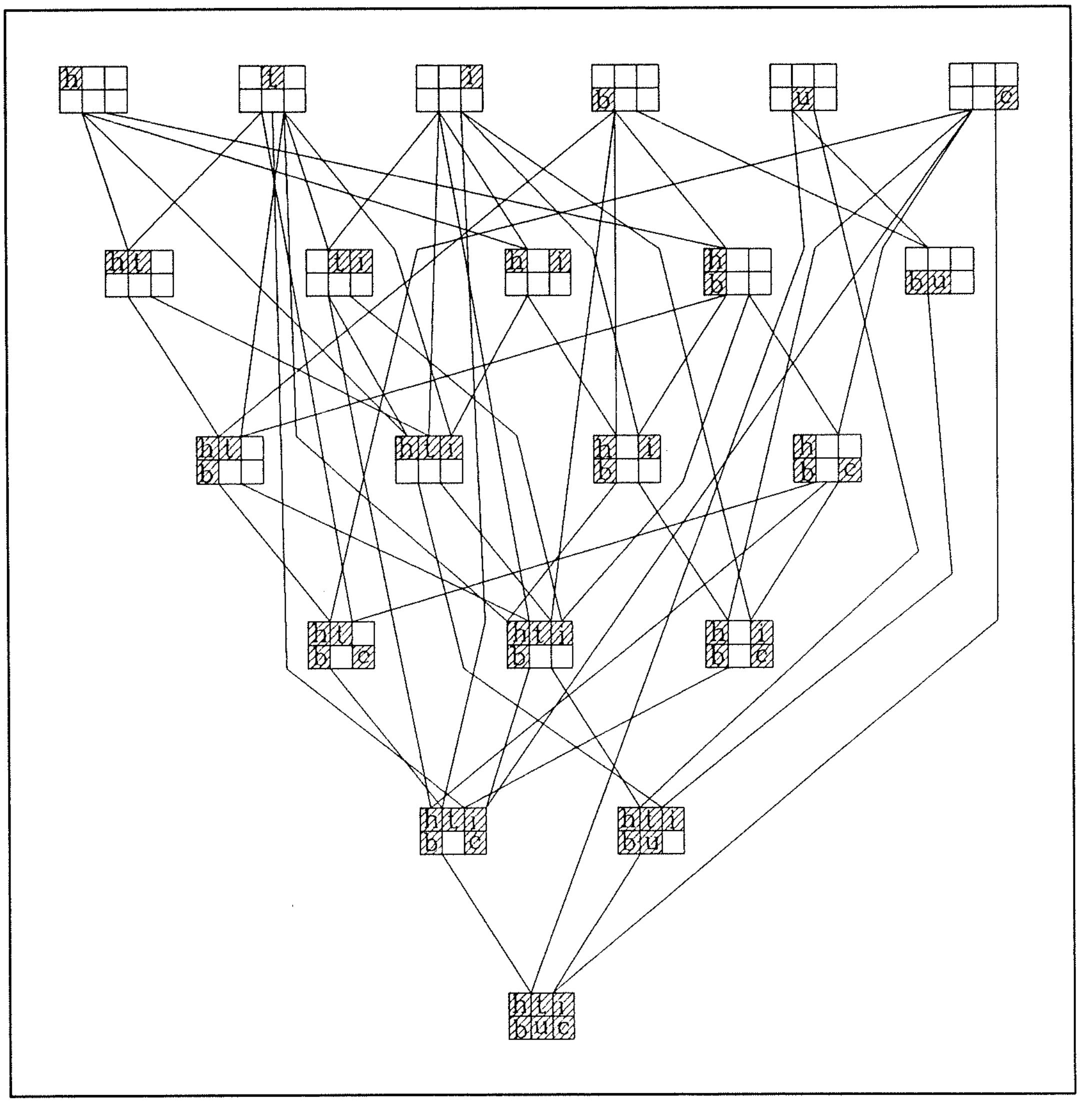


Figure 9.2. The ASG for the ball-point pen assembly

It can be observed that the total number of possible intermediate assembly states including the final assembly state is fifteen. That means the ASG has 21 nodes, fifteen nodes corresponding to each assembly state and six nodes for each individual component. The six nodes in the first level of the ASG represent one assembly state, i.e., the initial state or unassembled state. The ASG has 29 hyperarcs, one for each assembly task or state transition.

9.2.3 Sequence Selection

After all the feasible sequences are generated, the next step is to reduce the available sequence alternatives to identify the final choice. Of course, this selection process becomes simpler, if the user is able to identify some preferable assembly sequences. The following discussion will describe how various non-geometric constraints are applied to deduce the final sequence.

Constraint 1; Unwanted assembly states: While creating the geometric models of components, although ink is a liquid component, it was modeled as a solid component. Because of this assumption, some of the generated assembly sequences are not practicable. In practice, the ink has to be injected into the tube only after closing one end with the head. That means, the ink will go with the tube and head subassembly

Table 9.3. The AST for the ball-point assembly after imposing constraint 1

Level	Nodes (States)	Arcs (Tasks)
2	{h,t}	{(h), (t)}
	{h,b}	{(h), (b)}
	{b,u}	{(b), (u)}
3	{h,t,b}	{(h,t), (b)}; {(h,b), (t)}
	{h,t,i}	{(h,t), (i)}
	{h,b,c}	{(h,b), (c)}
4	{h,t,b,c}	{(h,t,b), (c}; {(h,b,c), (t)}
	{h,t,i,b}	{(h,t,b), (i)}; {(h,t,i), (b)}
5	{h,t,i,b,c}	{(h,t,b,c), (i)}; {(h,t,i,b), (c)}
	{h,t,i,b,u}	{(h,t,i,b), (u)}; {(h,t,i), (b,u)}
6	{h,t,i,b,u,c}	{(h,t,i,b,c), (u)}; {(h,t,i,b,u), (c)}

Total number of states = 11 Total number of tasks = 17

only. All the states that consist of ink without the tube and head will not be considered as good possibilities. In the ASG shown in Figure 9.2, the nodes {t,i}, {h,i}, {h,i,b} and {h,i,b,c} are not acceptable nodes and must be deleted, which in turn leads to the deletion of all the associated arcs. The remaining states and tasks are shown in Figure 9.3 and Table 9.3.

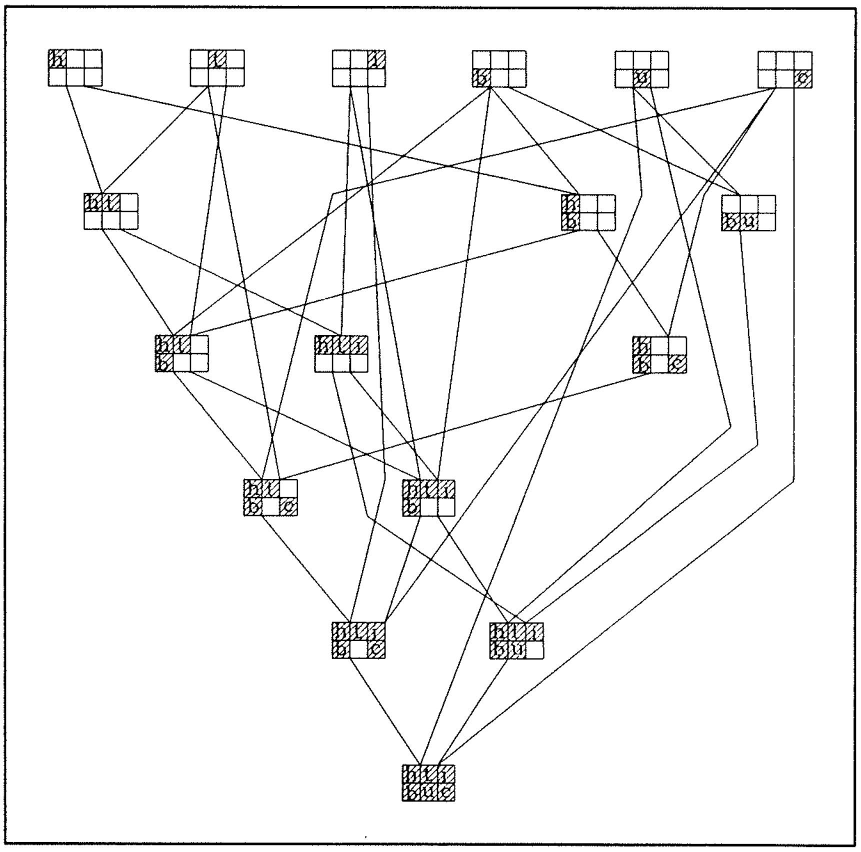


Figure 9.3. The ASG for the ball-point pen assembly after imposing constraint 1

Constraint 2; Difficult to do assembly task: If the body and head are assembled first, adding the tube to this subassembly and then injecting ink can be considered as an awkward assembly task. So, the assembly tasks {(h,t,b,c), (i)} and {(h,t,b), (i)} in which ink will be injected after head and tube are assembled to the body, are not desirable. Removal of the corresponding arcs from the ASG will leave the node {h,t,b,c} as dead. Again, deletion of this node will lead to the deletion of the arcs {(h,t,b), (c)}, and {(h,b,c), (t)} and subsequently the states {h,b,c}, {h,t,b} and {h,b}. Now there are only 7 assembly states left, which are formed through 9 assembly tasks representing 4 assembly sequences (or paths in the ASG) as shown in Figure 9.4. The equivalent AST is presented in Table 9.4.

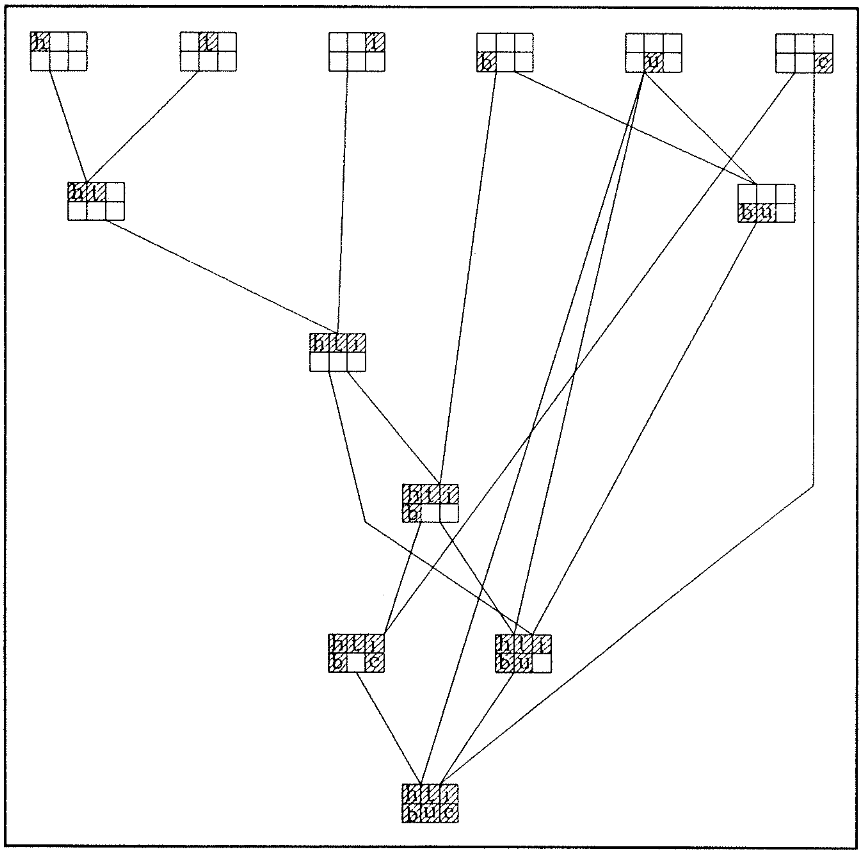


Figure 9.4. The ASG for the ball-point pen assembly after imposing constraint 2

Table 9.4. The AST for the ball-point pen assembly after imposing constraint 2

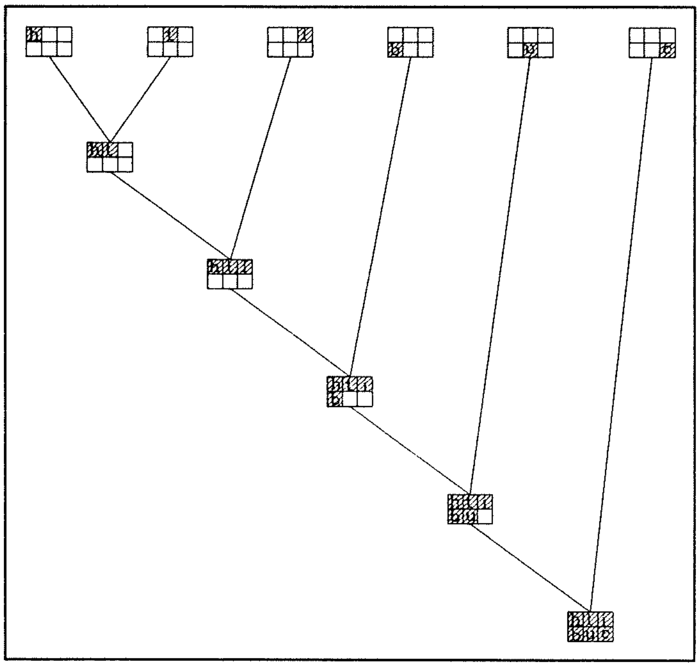
Level	Nodes (States)	Arcs (Tasks)
2	{h,t}	{(h), (t)}
	{b,u}	{(b), (u)}
3	{h,t,i}	{(h,t), (i)}
4	{h,t,i,b}	{(h,t,i), (b}
5	{h,t,i,b,c}	{(h,t,i,b), (c)}
	{h,t,i,b,u}	{(h,t,i,b), (u)}; {(h,t,i), (b,u)}
6	{h,t,i,b,u,c}	{(h,t,i,b,c), (u)}; {(h,t,i,b,u), (c)}

Total number of states = 7 Total number of tasks = 9

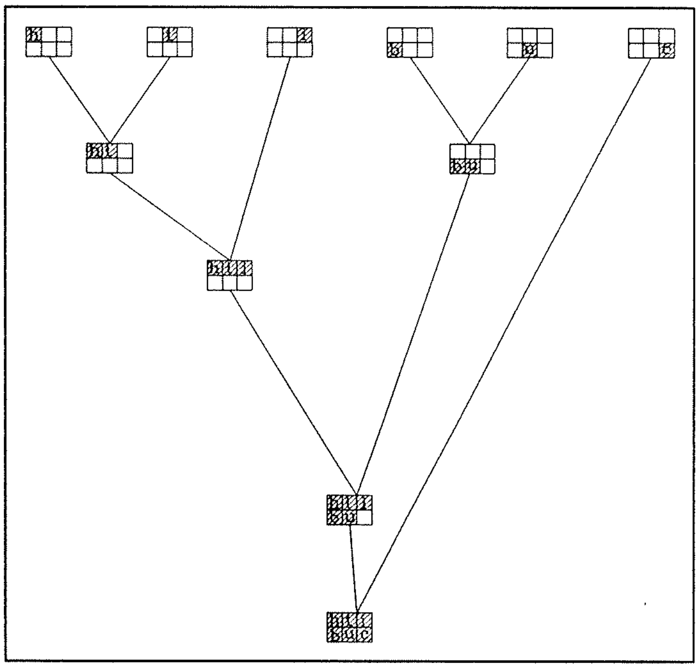
Constraint 3; Specified last component: Now, among the available four choices, the engineer needs to choose whether the cap or the button will be assembled last. Cap to body is a loose fit and button to body is a permanent fit. It is always preferable to assemble the button prior to the assembly of the cap, as it gives a more stable assembly (since it is a press-fit). With this constraint, only two sequences (or 6 states and 8 tasks) are left: one linear sequence (Figure 9.5a) and one sequence with subassemblies (Figure 9.5b). In actual practice, refills (head, tube and ink assembly) are made separately, because the same kind of refills will be used for a variety of pens. So the sequence in Figure 9.5b can be taken as the best sequence for this assembly. Thus, the developed methodology systematically comes up with the desired assembly sequence.

9.2.4 Comparison With Other Methods

To compare the proposed method with the other methods, this method does not need a seperate graphical representation to describe the relations among the components of the assembly. The relational information can be automatically deduced directly from



(a) Linear sequence



(b) Sequence with subassemblies

Figure 9.5. The final assembly sequence trees for the ball-point pen assembly

the design of the product. The liaison diagram will just give the contact information only, whereas the C and T functions provide not only contact information but also precedence information. The relational model presented here can include any other required geometric and non-geometric information, if necessary. The relational graph of Homem de Mello and Sanderson is too complex and it has several types of nodes and requires some expertise to construct the graph. Analysis of this relational graph is not simple as it requires the use of complex graph theory algorithms.

In liaison graph methods, even for the same set of components, for any change in the design, an entire question and answer session has to be repeated to extract the precedence relations. The situation is similar in the case of Homem de Mello and Sanderson's method and the entire cut-set decomposition of relational graph has to be repeated. But in the present method, updating of corresponding C and T functions will be sufficient to generate new assembly sequences. This could be done by simply performing the collision detection procedure for the pairs containing the parts that are modified.

Generation of assembly sequences from the relational knowledge in the present method is considered to be very systematic and simple as it deals with some binary information only. And it does not use any complex mathematics other than simple Boolean algebra principles. In addition to that, no questions are required to be answered. For this ball-point pen example, after the construction of liaison diagram, 64 questions are asked and answered in Bourjalut's method, whereas by De Fazio and Whitney's method, 10 questions which are more complex, are to be asked and answered.

When it comes to the sequence representation, inverted tree and LSG both give liaison sequences, but not the actual assembly sequences. Though, the LSG is supported with systematic editing features, it is not that simple to perform winnowing of the graph, as it does not provide a clear distinction of subassemblies.

Though the ASG and AND/OR graph have similarities, the later provides disassembly sequences. For the pen example, the ASG has more nodes than that of its AND/OR graph (Figure 2.7). This is again due to the assumption of ink as a solid component. But, the generation of the ASG is automatic, whereas in the construction of the AND/OR graph, the details are provided by the practitioner. In addition, the construction of the AND/OR graph is not as systematic as ASG. In ASG, the nodes are constructed systematically, from the lower levels to the higher levels and it is very well structured, since at any level L, the subassembly has L number of components. Finally, the ASG is supported with efficient editing features which is not the case with AND/OR graph. Thus, the ASG has the advantages of both AND/OR graph and LSG.

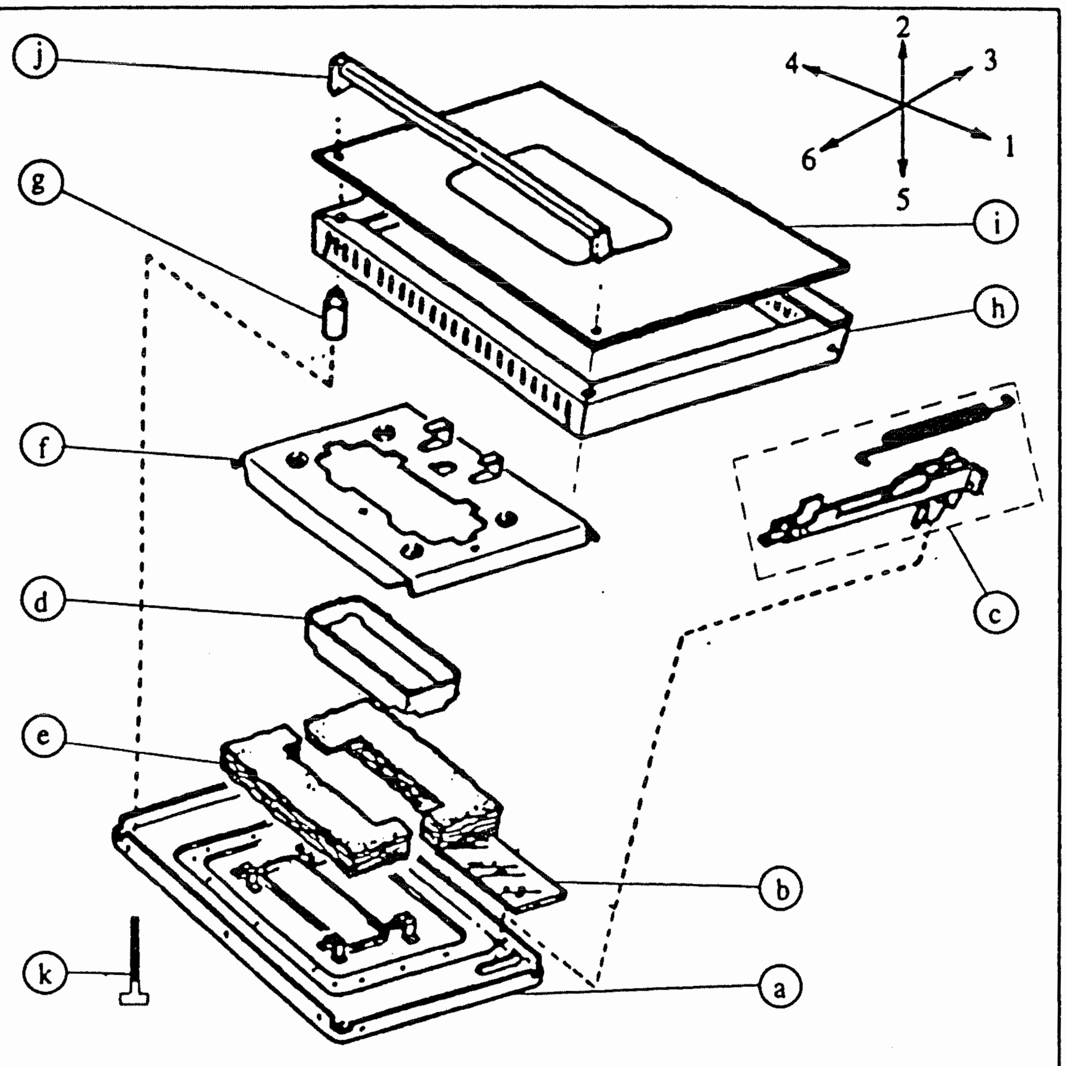
Another drawback of the liaison methods is that the number of liaisons may differ for minor modifications of the product design. If the number of liaisons is changed, the entire LSG has to be changed, because the number of cells in each box will be different, and so it is very difficult to estimate the complexity involved in the determination of sequences. But in our method, as long as the number of components is the same, the structure of the ASG will also be the same.

9.3 BUILT-IN OVEN DOOR ASSEMBLY

The second example considered here is a built-in oven door assembly (23) which represents a problem from the industry. Figure 9.6 shows an exploded view of all the components of the oven door assembly. The CAD models of the individual components have been created in their assembled position using PADL-2 and the display of its views is shown in Figure. 9.7. This design was taken from Frigidaire Canada Inc., a home appliances manufacturer. This product has been analyzed in order to demonstrate how the method we have developed can be applied to practical situations.

Table 9.5. Components of oven door assembly

CODE	NAME	QUANTITY	DIMENSIONS (inch)
a	Door body	1	22.50 x 18.40 x 2.0
b	Inner glass	1	13.50 x 6.50 x 0.134
c	Door hinge	2	16.75 x 1.75 x 4.75
d	Glass retainer	1	13.25 x 6.25 x 1.0
e	Insulation	1	17.5 x 15.0 x 2.0
f	Air wash panel	1	17.75 x 15.75 x 2.63
g	Spacer	2	1.75 x 0.50
h	Trim frame	1	22.75 x 18.65 x 1.50
i	Outer glass	1	22.75 x 18.56 x 0.12
j	Handle	1	21.80 x 2.0 x 0.90
k	Screw	2	2.12 x 0.40



a : Door body	g : Spacer
b : Inner glass	h : Trim frame
c : Door hinge	i : Outer glass
d : Glass retainer	J : Handle
e : Insulation	k : Screw
f : Air wash panel	

Figure 9.6. Exploded view of the oven door assembly

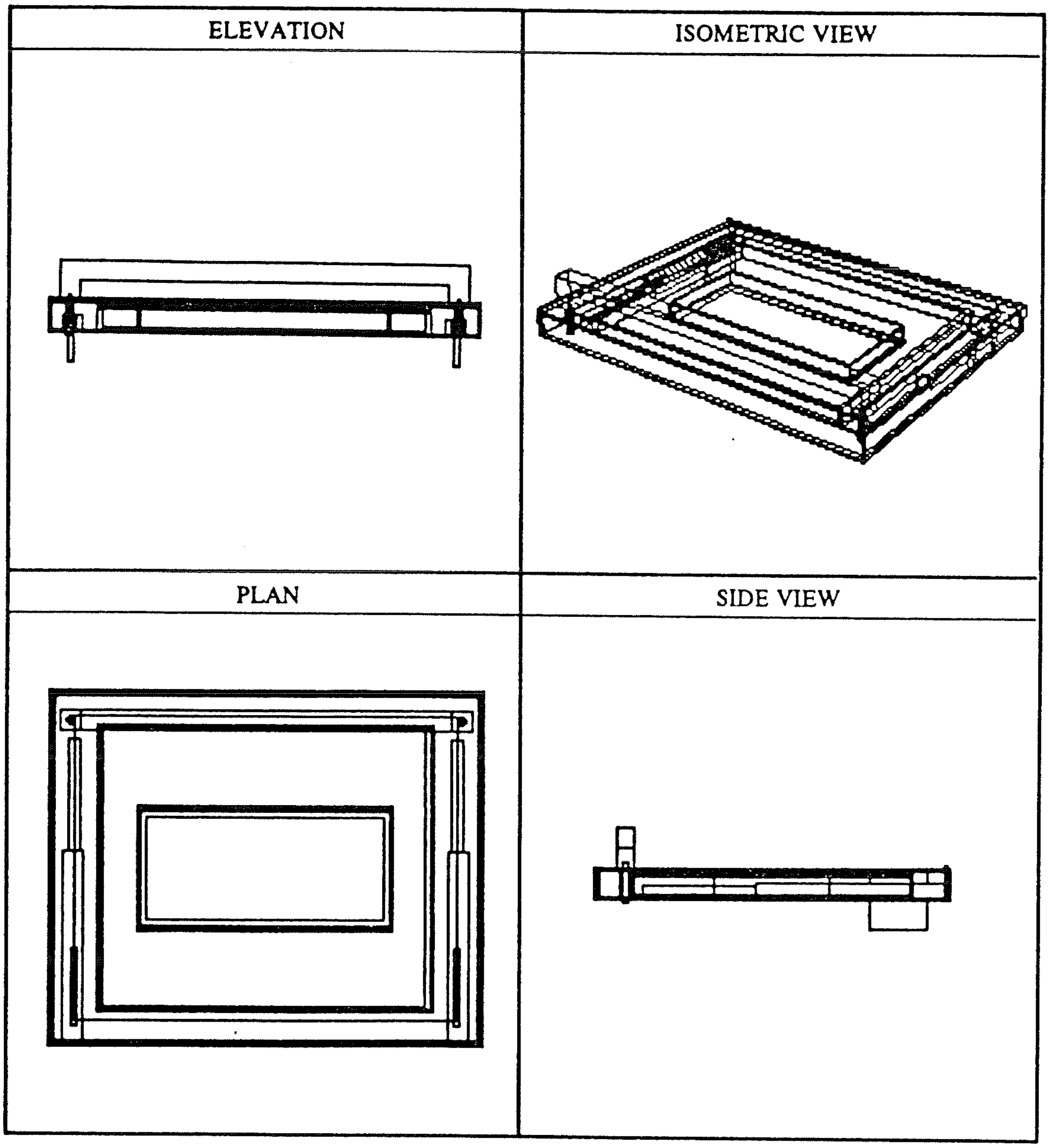


Figure 9.7. PADL-2 views of the built-in oven door assembly

This product consists of eleven basic component types. Table 9.5 gives the list of these components and quantity of each component in the assembly. For simplicity, each component has been given a unique code letter and these codes will be used in the sequence generation and representation. The fourth column of Table 9.5 gives the dimensions of a smallest envelope (a parallelepiped or a cylinder) that can encompass the component. The rotational type components are specified by two dimensions: length and diameter, whereas non-rotational components are specified by three dimensions: length, width and height.

9.3.1 Precedence Knowledge Extraction

By applying the collision detection algorithm for these components in their assembled position in the 3-D space, the resulting relational knowledge is presented in Table 9.6 in terms of C and T functions for all the pairs of components. It should be noted that, the dimensions of the insulation material are smaller in the assembled state than in the unassembled state as it gets compressed in the assembled state. Hence its dimensions in assembled state are used for precedence knowledge extraction.

In Table 9.6, there are a total of 110 ordered pairs. But it is not necessary to perform collision detection for all the ordered pairs, since 55 of them are complements to the other 55 pairs. For example, pair (b,a) is the complement of pair (a,b). Once the C and T functions of pair (a,b) are known, the same for the pair (b,a) can be easily obtained without performing the collision detection, as discussed in Chapter 4. The pairs in the right half of the Table 9.6 are the complements of the pairs in the left half.

Table 9.6. C & T functions for the oven door assembly

Pair	C-Function						T-Function						Pair	C-Function						T-Function					
	C_1	C_2	C_3	C_4	C_5	C_6	T_1	T_2	T_3	T_4	T_5	T_6		C_1	C_2	C_3	C_4	C_5	C_6	T_1	T_2	T_3	T_4	T_5	T_6
(a,b)	0	0	0	0	1	0	0	1	0	0	0	0	(b,a)	0	1	0	0	0	0	0	0	0	0	1	0
(a,c)	1	0	1	1	1	1	0	1	0	0	0	0	(c,a)	1	1	1	1	0	1	0	0	0	0	1	0
(a,d)	0	0	0	0	0	0	0	1	0	0	0	0	(d,a)	0	0	0	0	0	0	0	0	0	0	1	0
(a,e)	0	0	0	0	0	0	0	1	0	0	0	0	(e,a)	0	0	0	0	0	0	0	0	0	0	1	0
(a,f)	0	0	0	0	1	0	0	1	0	0	0	0	(f,a)	0	1	0	0	0	0	0	0	0	0	1	0
(a,g)	0	0	0	0	1	0	0	1	0	0	0	0	(g,a)	0	1	0	0	0	0	0	0	0	0	1	0
(a,h)	1	0	1	1	1	1	0	1	0	0	0	0	(h,a)	1	1	1	1	0	1	0	0	0	0	1	0
(a,i)	0	0	0	0	0	0	1	1	1	1	0	1	(i,a)	0	0	0	0	0	0	1	0	1	1	1	1
(a,j)	0	0	0	0	0	0	1	1	1	1	0	1	(j,a)	0	0	0	0	0	0	1	0	1	1	1	1
(a,k)	1	1	1	1	0	1	0	0	0	0	1	0	(k,a)	1	0	1	1	1	1	0	1	0	0	0	0
(b,c)	0	0	0	0	0	0	0	1	1	0	1	1	(c,b)	0	0	0	0	0	0	0	1	1	0	1	1
(b,d)	0	0	0	0	1	0	1	1	1	1	0	1	(d,b)	0	1	0	0	0	0	1	0	1	1	1	1
(b,e)	0	0	0	0	1	0	1	1	1	1	0	1	(e,b)	0	1	0	0	0	0	1	0	1	1	1	1
(b,f)	0	0	0	0	0	0	0	1	0	0	0	0	(f,b)	0	0	0	0	0	0	0	0	0	0	1	0
(b,g)	0	0	0	0	0	0	1	1	1	1	1	1	(g,b)	0	0	0	0	0	0	1	1	1	1	1	1
(b,h)	0	0	0	0	0	0	0	1	0	0	1	0	(h,b)	0	0	0	0	0	0	0	1	0	0	1	0
(b,i)	0	0	0	0	0	0	1	1	1	1	0	1	(i,b)	0	0	0	0	0	0	1	0	1	1	1	1
(b,j)	0	0	0	0	0	0	1	1	1	1	1	1	(j,b)	0	0	0	0	0	0	1	1	1	1	1	1
(b,k)	0	0	0	0	0	0	1	1	1	1	1	1	(k,b)	0	0	0	0	0	0	1	1	1	1	1	1
(c,d)	0	0	0	0	0	0	0	1	1	0	1	1	(d,c)	0	0	0	0	0	0	0	1	1	0	1	1
(c,e)	0	0	0	0	0	0	0	1	1	0	1	1	(e,c)	0	0	0	0	0	0	0	1	1	0	1	1
(c,f)	0	0	0	0	0	0	0	1	1	0	1	1	(f,c)	0	0	0	0	0	0	0	1	1	0	1	1
(c,g)	0	0	0	0	0	0	1	1	0	1	1	1	(g,c)	0	0	0	0	0	0	1	1	1	1	1	0
(c,h)	0	0	0	0	0	0	0	1	0	0	0	0	(h,c)	0	0	0	0	0	0	0	0	0	0	1	0
(c,i)	0	0	0	0	0	0	1	1	1	1	0	1	(i,c)	0	0	0	0	0	0	1	0	1	1	1	1
(c,j)	0	0	0	0	0	0	1	1	1	1	1	1	(j,c)	0	0	0	0	0	0	1	1	1	1	1	1
(c,k)	0	0	0	0	0	0	1	1	0	1	1	1	(k,c)	0	0	0	0	0	0	1	1	1	1	1	0

Table 9.6. (Continuation)

Pair	C-Function						T-Function						Pair	C-Function						T-Function					
	C_1	C_2	C_3	C_4	C_5	C_6	T_1	T_2	T_3	T_4	T_5	T_6		C_1	C_2	C_3	C_4	C_5	C_6	T_1	T_2	T_3	T_4	T_5	T_6
(d,e)	1	0	1	1	0	1	0	1	0	0	1	0	(e,d)	1	0	1	1	0	1	0	1	0	0	1	0
(d,f)	0	0	0	0	1	0	0	1	0	0	0	0	(f,d)	0	1	0	0	0	0	0	0	0	0	1	0
(d,g)	0	0	0	0	0	0	1	1	1	1	1	1	(g,d)	0	0	0	0	0	0	1	1	1	1	1	1
(d,h)	0	0	0	0	0	0	0	1	0	0	1	0	(h,d)	0	0	0	0	0	0	0	1	0	0	1	0
(d,i)	0	0	0	0	0	0	1	1	1	1	0	1	(i,d)	0	0	0	0	0	0	1	0	1	1	1	1
(d,j)	0	0	0	0	0	0	1	1	1	1	1	1	(j,d)	0	0	0	0	0	0	1	1	1	1	1	1
(d,k)	0	0	0	0	0	0	1	1	1	1	1	1	(k,d)	0	0	0	0	0	0	1	1	1	1	1	1
(e,f)	0	0	0	1	0	0	0	1	0	0	0	0	(f,e)	0	0	0	1	0	0	0	0	0	0	1	0
(e,g)	0	0	0	0	0	0	1	1	1	1	1	1	(g,e)	0	0	0	0	0	0	1	1	1	1	1	1
(e,h)	0	0	0	0	0	0	0	1	0	0	1	0	(h,e)	0	0	0	0	0	0	0	1	0	0	1	0
(e,i)	0	0	0	0	0	0	1	1	1	1	0	1	(i,e)	0	0	0	0	0	0	1	0	1	1	1	1
(e,j)	0	0	0	0	0	0	1	1	1	1	1	1	(j,e)	0	0	0	0	0	0	1	1	1	1	1	1
(e,k)	0	0	0	0	0	0	1	1	1	1	1	1	(k,e)	0	0	0	0	0	0	1	1	1	1	1	1
(f,g)	0	0	0	0	0	0	1	1	1	1	1	1	(g,f)	0	0	0	0	0	0	1	1	1	1	1	1
(f,h)	0	0	0	0	0	0	0	1	0	0	1	0	(h,f)	0	0	0	0	0	0	0	1	0	0	1	0
(f,i)	0	0	0	0	0	0	1	1	1	1	0	1	(i,f)	0	0	0	0	0	0	1	0	1	1	1	1
(f,j)	0	0	0	0	0	0	1	1	1	1	1	1	(j,f)	0	0	0	0	0	0	1	1	1	1	1	1
(f,k)	0	0	0	0	0	0	1	1	1	1	1	1	(k,f)	0	0	0	0	0	0	1	1	1	1	1	1
(g,h)	1	0	1	1	1	1	0	1	0	0	0	0	(h,g)	1	1	1	1	0	1	0	0	0	0	1	0
(g,i)	0	0	0	0	1	0	1	1	1	1	0	1	(i,g)	0	1	0	0	0	0	1	0	1	1	1	1
(g,j)	0	0	0	0	0	0	1	1	1	1	0	1	(j,g)	0	0	0	0	0	0	1	0	1	1	1	1
(g,k)	1	1	1	1	0	1	0	0	0	0	1	0	(k,g)	1	0	1	1	1	1	0	1	0	0	0	0
(h,i)	0	1	1	0	1	0	1	0	0	1	0	1	(i,h)	0	1	0	0	1	1	0	1	1	0	1	0
(h,j)	0	0	0	0	0	0	1	1	1	1	0	1	(j,h)	0	0	0	0	0	0	0	0	0	0	0	0
(h,k)	0	0	0	0	0	0	0	0	0	0	1	0	(k,h)	0	0	0	0	0	0	0	0	0	0	0	0
(i,j)	0	0	0	0	1	0	1	1	1	1	0	1	(j,i)	0	1	0	0	0	0	0	0	0	0	1	0
(i,k)	1	0	1	1	0	1	0	0	0	0	1	0	(k,i)	1	0	1	1	1	1	1	0	1	1	0	1
(j,k)	1	1	1	1	0	1	0	0	0	0	1	0	(k,j)	1	0	1	1	1	1	1	1	1	1	0	1

9.3.2. Sequence Generation

The information in the Table 9.6 is given as input to the sequence generation algorithm, and this algorithm produced only the feasible assembly states through which the final assembly sequence can be obtained.

In this example, the initial state is { {a}, {b}, {c}, {d}, {e}, {f}, {g}, {h}, {i}, {j}, {k}}. Since there are 11 components in the assembly, there will be 11 nodes in the first level of the ASG. The final state or N^{th} level node is {a, b, c, d, e, f, g, h, i, j, k}. As discussed in Chapter 6, for assemblies with large number of components, the ASG will become cumbersome. Hence, initially only the table representation is used to represent the sequences. The AST shown in Table 9.7 represents all the geometrically feasible sequences. Since the initial state (first level) corresponds to the individual components and is not the result of any assembly tasks, it is not presented in the table. Moreover, this level will not be changed even after applying the selection criteria. From Table 9.7, it is observed that the given assembly has 71 assembly states including the final assembly state and these states are formed by 246 assembly tasks.

9.3.3 Sequence Selection

The generation of all feasible assembly sequences as demonstrated does not fulfill the objective of this research which is to determine the best assembly sequence. Moreover, some of the generated assembly states and tasks are not desirable. However, in order that the technique be immediately useful for practical application in the industry, an engineer must be in a position to reduce the number of sequences to a relatively small number of choices by casting out the awkward ones and retaining only those that are practically convenient. Sweeping reductions of this sort are possible in

Table 9.7. The AST for the oven-door assembly

Level	Nodes (States)	Arcs (Tasks)
2	{a,b}	{(a), (b)}
	{a,c}	{(a), (c)}
	{a,g}	{(a), (g)}
	{a,k}	{(a), (k)}
	{b,d}	{(b), (d)}
	{b,e}	{(b), (e)}
	{d,e}	{(d), (e)}
	{d,f}	{(d), (f)}
	{e,f}	{(e), (f)}
	{g,h}	{(g), (h)}
	{h,i}	{(h), (i)}
	{i,j}	{(i), (j)}
3	{a,b,c}	{(a,b), (c)}; {(a,c), (b)}
	{a,b,d}	{(a,b), (d)}
	{a,b,e}	{(a,b), (e)}; {(b,e), (a)}
	{a,b,g}	{(a,b), (g)}; {(a,g), (b)}
	{a,b,k}	{(a,b), (k)}; {(a,k), (b)}
	{a,c,g}	{(a,c), (g)}; {(a,g), (c)}
	{a,c,k}	{(a,c), (k)}; {(a,k), (c)}
	{a,g,k}	{(a,g), (k)}; {(a,k), (g)}
	{b,d,e}	{(b,d), (e)}; {(b,e), (d)}; {(d,e), (b)}
	{d,e,f}	{(d,e), (f)}; {(d,f), (e)}; {(e,f), (d)}
	{g,h,i}	{(g,h), (i)}; {(h,i), (g)}
	{h,i,j}	(h,i), (j)}; {(i,j), (h)}
4	{a,b,c,d}	{(a,b,c), (d)}; {(a,b,d), (c)}; {(a,c), (b,d)}
	{a,b,c,e}	{(a,b,c), (e)}; {(a,b,e), (c)}; {(a,c), (b,e)}
	{a,b,c,g}	{(a,b,c), (g)}; {(a,b,g), (c)}; {(a,c,g), (b)}
	{a,b,c,k}	{(a,b,c), (k)}; {(a,b,k), (c)}; {(a,c,k), (b)}

Table 9.7. (Continuation)

Level	Nodes (States)	Arcs (Tasks)
4	{a,b,d,e}	{(a,b,d), (e)}; {(a,b,e), (d)}; {(b,d,e), (a)} {(a,b), (d,e)}
	{a,b,d,g}	{(a,b,d), (g)}; {(a,b,g), (d)}; {(a,g), (b,d)}
	{a,b,d,k}	{(a,b,d), (k)}; {(a,b,k), (d)}; {(a,k), (b,d)}
	{a,b,e,g}	{(a,b,e), (g)}; {(a,b,g), (e)}; {(a,g), (b,e)}
	{a,b,e,k}	{(a,b,e), (k)}; {(a,b,k), (e)}; {(a,k), (b,e)}
	{a,b,g,k}	{(a,b,g), (k)}; {(a,b,k), (g)}; {(a,g,k), (b)}
	{a,c,g,h}	{(a,c,g), (h)}; {(a,c), (g,h)}
	{a,c,g,k}	{(a,c,g), (k)}; {(a,c,k), (g)}; {(a,g,k), (c)}
	{b,d,e,f}	{(b,d,e), (f)}; {(d,e,f), (b)} {(b,d), (e,f)}; {(b,e), (d,f)}
	{g,h,i,j}	{(g,h,i), (j)}; {(h,i,j), (g)}; {(g,h), (i,j)}
5	{a,b,c,d,e}	{(a,b,c,d), (e)}; {(a,b,c,e), (d)}; {(a,b,d,e), (c)} {(a,b,c), (d,e)}; {(b,d,e), (a,c)}
	{a,b,c,d,g}	{(a,b,c,d), (g)}; {(a,b,c,g), (d)}; {(a,b,d,g), (c)} {(a,c,g), (b,d)}
	{a,b,c,d,k}	{(a,b,c,d), (k)}; {(a,b,c,k), (d)}; {(a,b,d,k), (c)} {(a,c,k), (b,d)}
	{a,b,c,e,g}	{(a,b,c,e), (g)}; {(a,b,c,g), (e)}; {(a,b,e,g), (c)} {(a,c,g), (b,e)}
	{a,b,c,e,k}	{(a,b,c,e), (k)}; {(a,b,c,k), (e)}; {(a,b,e,k), (c)} {(a,c,k), (b,e)}
	{a,b,c,g,h}	{(a,b,c,g), (h)}; {(a,c,g,h), (b)}; {(a,b,c), (g,h)}
	{a,b,c,g,k}	{(a,b,c,g), (k)}; {(a,b,c,k), (g)}; {(a,b,g,k), (c)} {(a,c,g,k), (b)}
	{a,b,d,e,f}	{(a,b,d,e), (f)}; {(b,d,e,f), (a)} {(a,b,d), (e,f)}; {(a,b,e), (d,f)}; {(d,e,f), (a,b)}
	{a,b,d,e,g}	{(a,b,d,e), (g)}; {(a,b,d,g), (e)}; {(a,b,e,g), (d)} {(a,b,g), (d,e)}; {(b,d,e), (a,g)}
	{a,b,d,e,k}	{(a,b,d,e), (k)}; {(a,b,d,k), (e)}; {(a,b,e,k), (d)} {(a,b,k), (d,e)}; {(b,d,e), (a,k)}
	{a,b,d,g,k}	{(a,b,d,g), (k)}; {(a,b,d,k), (g)}; {(a,b,g,k), (d)} {(a,g,k), (b,d)}
	{a,b,e,g,k}	{(a,b,e,g), (k)}; {(a,b,e,k), (g)}; {a,g,k}, {b,e}

Table 9.7. (Continuation)

Level	Nodes (States)	Arcs (Tasks)
6	{a,b,c,d,e,f}	{(a,b,c,d,e), (f)}; {(a,b,d,e,f), (c)}; {(a,b,c,d), (e,f)} {(a,b,c,e), (d,f)}; {(b,d,e,f), (a,c)}; {(a,b,c), (d,e,f)}
	{a,b,c,d,e,g}	{(a,b,c,d,e), (g)}; {(a,b,c,d,g), (e)}; {(a,b,c,e,g), (d)} {(a,b,d,e,g), (c)}; {(a,b,c,g), (d,e)}; {(a,c,g), (b,d,e)}
	{a,b,c,d,e,k}	{(a,b,c,d,e), (k)}; {(a,b,c,d,k), (e)}; {(a,b,c,e,k), (d)} {(a,b,d,e,k), (c)}; {(a,b,c,k), (d,e)}; {(a,c,k), (b,d,e)}
	{a,b,c,d,g,h}	{(a,b,c,d,g), (h)}; {(a,b,c,g,h), (d)} {(a,b,c,d), (g,h)}; {(a,c,g,h), (b,d)}
	{a,b,c,d,g,k}	{(a,b,c,d,g), (k)}; {(a,b,c,d,k), (g)}; {(a,b,c,g,k), (d)} {(a,b,d,g,k), (c)}; {(a,c,g,k), (b,d)}
	{a,b,c,e,g,h}	{(a,b,c,e,g), (h)}; {(a,b,c,g,h), (e)} {(a,b,c,e), (g,h)}; {(a,c,g,h), (b,e)}
	{a,b,c,e,g,k}	{(a,b,c,e,g), (k)}; {(a,b,c,e,k), (g)}; {(a,b,c,g,k), (e)} {(a,b,e,g,k), (c)}; {(a,c,g,k), (b,e)}
	{a,b,d,e,f,g}	{(a,b,d,e,f), (g)}; {(a,b,d,e,g), (f)}; {(a,b,d,g), (e,f)} {(a,b,e,g), (d,f)}; {(b,d,e,f), (a,g)}; {(a,b,g), (d,e,f)}
	{a,b,d,e,f,k}	{(a,b,d,e,f), (k)}; {(a,b,d,e,k), (f)}; {(a,b,d,k), (e,f)} {(a,b,e,k), (d,f)}; {(b,d,e,f), (a,k)}; {(a,b,k), (d,e,f)}
	{a,b,d,e,g,k}	{(a,b,d,e,g), (k)}; {(a,b,d,e,k), (g)}; {(a,b,d,g,k), (e)} {(a,b,e,g,k), (d)}; {(a,b,g,k), (d,e)}; {(a,g,k), (b,d,e)}
7	{a,b,c,d,e,f,g}	{(a,b,c,d,e,f), (g)}; {(a,b,c,d,e,g), (f)}; {(a,b,d,e,f,g), (c)} {(a,b,c,d,g), (e,f)}; {(a,b,c,e,g), (d,f)}; {(a,b,c,g), (d,e,f)} {(b,d,e,f), (a,c,g)}
	{a,b,c,d,e,f,k}	{(a,b,c,d,e,f), (k)}; {(a,b,c,d,e,k), (f)}; {(a,b,d,e,f,k), (c)} {(a,b,c,d,k), (e,f)}; {(a,b,c,e,k), (d,f)}; {(a,b,c,k), (d,e,f)} {(b,d,e,f), (a,c,k)}
	{a,b,c,d,e,g,h}	{(a,b,c,d,e,g), (h)}; {(a,b,c,d,g,h), (e)} {(a,b,c,e,g,h), (d)}; {(a,b,c,d,e), (g,h)} {(a,b,c,g,h), (d,e)}; {(a,c,g,h), (b,d,e)}
	{a,b,c,d,e,g,k}	{(a,b,c,d,e,g), (k)}; {(a,b,c,d,e,k), (g)}; {(a,b,c,d,g,k), (e)} {(a,b,c,e,g,k), (d)}; {(a,b,d,e,g,k), (c)}; {(a,b,c,g,k), (d,e)} {(a,c,g,k), (b,d,e)}
	{a,b,d,e,f,g,k}	{(a,b,d,e,f,g), (k)}; {(a,b,d,e,f,k), (g)}; {(a,b,d,e,g,k), (f)} {(a,b,d,g,k), (e,f)}; {(a,b,e,g,k), (d,f)}; {(a,b,g,k), (d,e,f)} {(b,d,e,f), (a,g,k)}

Table 9.7. (Continuation)

Level	Nodes (States)	Arcs (Tasks)
8	{a,b,c,d,e,f,g,h}	{(a,b,c,d,e,f,g), (h)}; {(a,b,c,d,e,g,h), (f)} {(a,b,c,d,e,f), (g,h)}; {(a,b,c,d,g,h), (e,f)} {(a,b,c,e,g,h), (d,f)}; {(a,b,c,g,h), (d,e,f)} {(a,c,g,h), (b,d,e,f)}
	(a,b,c,d,e,f,g,k}	{(a,b,c,d,e,f,g), (k)}; {(a,b,c,d,e,f,k), (g)} {(a,b,c,d,e,g,k), (f)}; {(a,b,d,e,f,g,k), (c)} {(a,b,c,d,g,k), (e,f)}; {(a,b,c,e,g,k), (d,f)} {(a,b,c,g,k), (d,e,f)}; {(a,c,g,k), (b,d,e,f)}
9	{a,b,c,d,e,f,g,h,i}	{(a,b,c,d,e,f,g,h), (i)}; {(a,b,c,d,e,f,g), (h,i)} {(a,b,c,d,e,f), (g,h,i)}
10	{a,b,c,d,e,f,g,h,i,j}	{(a,b,c,d,e,f,g,h,i), (j)}; {(a,b,c,d,e,f,g,h), (i,j)} {(a,b,c,d,e,f,g), (h,i,j)}; {(a,b,c,d,e,f), (g,h,i,j)}
	{a,b,c,d,e,f,g,h,i,k}	{(a,b,c,d,e,f,g,h,i), (k)}; {(a,b,c,d,e,f,g,k), (h,i)} {(a,b,c,d,e,f,k), (g,h,i)}
11	{a,b,c,d,e,f,g,h,i,j,k}	{(a,b,c,d,e,f,g,h,i,j), (k)}; {(a,b,c,d,e,f,g,h,i,k), (j)} {(a,b,c,d,e,f,g,k), (h,i,j)}; {(a,b,c,d,e,f,k), (g,h,i,j)}

Total number of states = 71 Total number of tasks = 246

this example. The following discussion will show how various selection criteria are applied to reduce the number of sequences for further detailed analysis.

Constraint 1; Specified last component: Functionally, the attachment agents should be assembled after the primary components are already assembled. In this example, the screw (k) is an attachment agent, and it is the one that actually keeps all the components together. This component must be assembled at the end. So all the intermediate subassemblies with component k already installed are not desirable. Simple scanning through the second column of the Table 9.7 indicates that there are 26 assembly states comprising component k. All these states and tasks through which these are formed, except the last one in the 11th level, must be deleted. This can easily be performed by deleting the corresponding rows in the Table 9.7. In the last level,

Table 9.8. The AST for the oven door assembly after imposing constraint 1

Level	Nodes (States)	Arcs (Tasks)
2	{a,b}	{(a), (b)}
	{a,c}	{(a), (c)}
	{a,g}	{(a), (g)}
	{b,d}	{(b), (d)}
	{b,e}	{(b), (e)}
	{d,e}	{(d), (e)}
	{d,f}	{(d), (f)}
	{e,f}	{(e), (f)}
	{g,h}	{(g), (h)}
	{h,i}	{(h), (i)}
	{i,j}	{(i), (j)}
3	{a,b,c}	{(a,b), (c)}; {(a,c), (b)}
	{a,b,d}	{(a,b), (d)}
	{a,b,e}	{(a,b), (e)}; {(b,e), (a)}
	{a,b,g}	{(a,b), (g)}; {(a,g), (b)}
	{a,c,g}	{(a,c), (g)}; {(a,g), (c)}
	{b,d,e}	{(b,d), (e)}; {(b,e), (d)}; {(d,e), (b)}
	{d,e,f}	{(d,e), (f)}; {(d,f), (e)}; {(e,f), (d)}
	{g,h,i}	{(g,h), (i)}; {(h,i), (g)}
	{h,i,j}	{(h,i), (j)}; {(i,j), (h)}
4	{a,b,c,d}	{(a,b,c), (d)}; {(a,b,d), (c)}; {(a,c), (b,d)}
	{a,b,c,e}	{(a,b,c), (e)}; {(a,b,e), (c)}; {(a,c), (b,e)}
	{a,b,c,g}	{(a,b,c), (g)}; {(a,b,g), (c)}; {(a,c,g), (b)}
	{a,b,d,e}	{(a,b,d), (e)}; {(a,b,e), (d)}; {(b,d,e), (a)}; {(a,b), (d,e)}
	{a,b,d,g}	{(a,b,d), (g)}; {(a,b,g), (d)}; {(a,g), (b,d)}
	{a,b,e,g}	{(a,b,e), (g)}; {(a,b,g), (e)}; {(a,g), (b,e)}
	{a,c,g,h}	{(a,c,g), (h)}; {(a,c), (g,h)}
	{b,d,e,f}	{(b,d,e), (f)}; {(d,e,f), (b)}; {(b,d), (e,f)}; {(b,e), (d,f)}
	{g,h,i,j}	{(g,h,i), (j)}; {(h,i,j), (g)}; {(g,h), (i,j)}

Table 9.8. (Continuation)

Level	Nodes (States)	Arcs (Tasks)

5	{a,b,c,d,e}	{(a,b,c,d), (e)}; {(a,b,c,e), (d)}; {(a,b,d,e), (c)}
		{(a,b,c), (d,e)}; {(b,d,e), (a,c)}

	{a,b,c,d,g}	{(a,b,c,d), (g)}; {(a,b,c,g), (d)}; {(a,b,d,g), (c)}
		{(a,c,g), (b,d)}

	{a,b,c,e,g}	{(a,b,c,e), (g)}; {(a,b,c,g), (e)}; {(a,b,e,g), (c)}
		{(a,c,g), (b,e)}

	{a,b,c,g,h}	{(a,b,c,g), (h)}; {(a,c,g,h), (b)}; {(a,b,c), (g,h)}

	{a,b,d,e,f}	{(a,b,d,e), (f)}; {(b,d,e,f), (a)}
		{(a,b,d), (e,f)}; {(a,b,e), (d,f)}; {(d,e,f), (a,b)}

	{a,b,d,e,g}	{(a,b,d,e), (g)}; {(a,b,d,g), (e)}; {(a,b,e,g), (d)}
		{(a,b,g), (d,e)}; {(b,d,e), (a,g)}

6	{a,b,c,d,e,f}	{(a,b,c,d,e), (f)}; {(a,b,d,e,f), (c)}; {(a,b,c,d), (e,f)}
		{(a,b,c,e), (d,f)}; {(b,d,e,f), (a,c)}; {(a,b,c), (d,e,f)}

	{a,b,c,d,e,g}	{(a,b,c,d,e), (g)}; {(a,b,c,d,g), (e)}; {(a,b,c,e,g), (d)}
		{(a,b,d,e,g), (c)}; {(a,b,c,g), (d,e)}; {(a,c,g), (b,d,e)}

	{a,b,c,d,g,h}	{(a,b,c,d,g), (h)}; {(a,b,c,g,h), (d)}
		{(a,b,c,d), (g,h)}; {(a,c,g,h), (b,d)}

	{a,b,c,e,g,h}	{(a,b,c,e,g), (h)}; {(a,b,c,g,h), (e)}
		{(a,b,c,e), (g,h)}; {(a,c,g,h), (b,e)}

	{a,b,d,e,f,g}	{(a,b,d,e,f), (g)}; {(a,b,d,e,g), (f)}; {(a,b,d,g), (e,f)}
		{(a,b,e,g), (d,f)}; {(b,d,e,f), (a,g)}; {(a,b,g), (d,e,f)}

7	{a,b,c,d,e,f,g}	{(a,b,c,d,e,f), (g)}; {(a,b,c,d,e,g), (f)}
		{(a,b,d,e,f,g), (c)}; {(a,b,c,d,g), (e,f)}
		{(a,b,c,e,g), (d,f)}; {(a,b,c,g), (d,e,f)}
		{(b,d,e,f), (a,c,g)}

	{a,b,c,d,e,g,h}	{(a,b,c,d,e,g), (h)}; .{(a,b,c,d,g,h), (e)}
		{(a,b,c,e,g,h), (d)};..{(a,b,c,d,e), (g,h)}
		{(a,b,c,g,h), (d,e)}; .{(a,c,g,h), (b,d,e)}

8	{a,b,c,d,e,f,g,h}	{(a,b,c,d,e,f,g), (h)}; {(a,b,c,d,e,g,h), (f)}
		{(a,b,c,d,e,f), (g,h)}; {(a,b,c,d,g,h), (e,f)}
		{(a,b,c,e,g,h), (d,f)}; {(a,b,c,g,h), (d,e,f)}
		{(a,c,g,h), (b,d,e,f)}

9	{a,b,c,d,e,f,g,h,i}	{(a,b,c,d,e,f,g,h), (i)}; {(a,b,c,d,e,f,g), (h,i)}
		{(a,b,c,d,e,f), (g,h,i)}

10	{a,b,c,d,e,f,g,h,i,j}	{(a,b,c,d,e,f,g,h,i), (j)}; {(a,b,c,d,e,f,g,h), (i,j)}
		{(a,b,c,d,e,f,g), (h,i,j)}; {(a,b,c,d,e,f), (g,h,i,j)}

11	{a,b,c,d,e,f,g,h,i,j,k}	{(a,b,c,d,e,f,g,h,i,j), (k)}

Total number of states = 46 Total number of tasks = 138

the assembly is formed through 4 different assembly tasks. Of these, three tasks involve the subassemblies with component k, which must also be deleted. The resulting AST with 46 states and 138 tasks is shown in Table 9.8. It shows that this constraint has reduced the number of states and tasks considerably.

Constraint 2; Linear sequences: The door assembly itself is a subassembly in the entire oven assembly. Actually at Frigidaire, this subassembly is assembled on a separate line and then it is added to the oven body. Considering the assembly line for the entire oven, it already has some modularity in it. By introducing more modularity

Table 9.9. The AST for the over door assembly after imposing constraint 2

Level	Nodes (States)	Arcs (Tasks)
2	{a,b}	{(a), (b)}
	{a,c}	{(a), (c)}
	{a,g}	{(a), (g)}
	{b,d}	{(b), (d)}
	{b,e}	{(b), (e)}
	{d,e}	{(d), (e)}
	{d,f}	{(d), (f)}
	{e,f}	{(e), (f)}
3	{a,b,c}	{(a,b), (c)}; {(a,c), (b)}
	{a,b,d}	{(a,b), (d)}
	{a,b,e}	{(a,b), (e)}; {(b,e), (a)}
	{a,b,g}	{(a,b), (g)}; {(a,g), (b)}
	{a,c,g}	{(a,c), (g)}; {(a,g), (c)}
	{b,d,e}	{(b,d), (e)}; {(b,e), (d)}; {(d,e), (b)}
	{d,e,f}	{(d,e), (f)}; {(d,f), (e)}; {(e,f), (d)}

Table 9.9. (Continuation)

Level	Nodes (States)	Arcs (Tasks)
4	{a,b,c,d}	{(a,b,c), (d)}; {(a,b,d), (c)}
	{a,b,c,e}	{(a,b,c), (e)}; {(a,b,e), (c)}
	{a,b,c,g}	{(a,b,c), (g)}; {(a,b,g), (c)}; {(a,c,g), (b)}
	{a,b,d,e}	{(a,b,d), (e)}; {(a,b,e), (d)}; {(b,d,e), (a)}
	{a,b,d,g}	{(a,b,d), (g)}; {(a,b,g), (d)}
	{a,b,e,g}	{(a,b,e), (g)}; {(a,b,g), (e)}
	{a,c,g,h}	{(a,c,g), (h)}
	{b,d,e,f}	{(b,d,e), (f)}; {(d,e,f), (b)}
5	{a,b,c,d,e}	{(a,b,c,d), (e)}; {(a,b,c,e), (d)}; {(a,b,d,e), (c)}
	{a,b,c,d,g}	{(a,b,c,d), (g)}; {(a,b,c,g), (d)}; {(a,b,d,g), (c)}
	{a,b,c,e,g}	{(a,b,c,e), (g)}; {(a,b,c,g), (e)}; {(a,b,e,g), (c)}
	{a,b,c,g,h}	{(a,b,c,g), (h)}; {(a,c,g,h), (b)}
	{a,b,d,e,f}	{(a,b,d,e), (f)}; {(b,d,e,f), (a)}
	{a,b,d,e,g}	{(a,b,d,e), (g)}; {(a,b,d,g), (e)}; {(a,b,e,g), (d)}
6	{a,b,c,d,e,f}	{(a,b,c,d,e), (f)}; {(a,b,d,e,f), (c)}
	{a,b,c,d,e,g}	{(a,b,c,d,e), (g)}; {(a,b,c,d,g), (e)} {(a,b,c,e,g), (d)} {(a,b,d,e,g), (c)}
	{a,b,c,d,g,h}	{(a,b,c,d,g), (h)}; {(a,b,c,g,h), (d)}
	{a,b,c,e,g,h}	{(a,b,c,e,g), (h)}; {(a,b,c,g,h), (e)}
	{a,b,d,e,f,g}	{(a,b,d,e,f), (g)}; {(a,b,d,e,g), (f)}
7	{a,b,c,d,e,f,g}	{(a,b,c,d,e,f), (g)}; {(a,b,c,d,e,g), (f)} {(a,b,d,e,f,g), (c)}
	{a,b,c,d,e,g,h}	{(a,b,c,d,e,g), (h)}; {(a,b,c,d,g,h), (e)} {(a,b,c,e,g,h), (d)}
8	{a,b,c,d,e,f,g,h}	{(a,b,c,d,e,f,g), (h)}; {(a,b,c,d,e,g,h), (f)}
9	{a,b,c,d,e,f,g,h,i}	{(a,b,c,d,e,f,g,h), (i)}
10	{a,b,c,d,e,f,g,h,i,j}	{(a,b,c,d,e,f,g,h,i), (j)}
11	{a,b,c,d,e,f,g,h,i,j,k}	{(a,b,c,d,e,f,g,h,i,j), (k)}

Total number of states = 40 Total number of tasks = 79

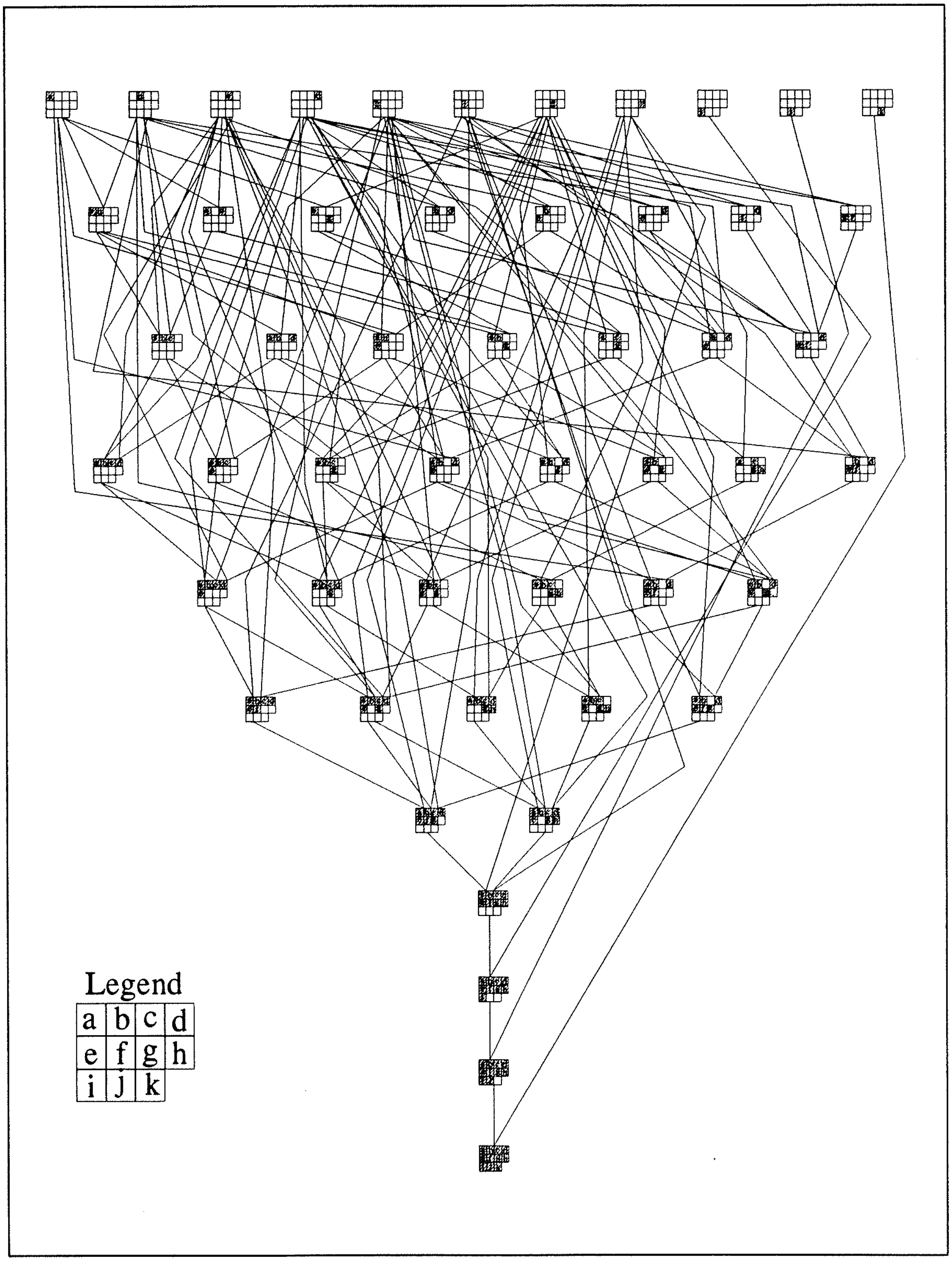


Figure 9.8. The ASG for the oven door assembly after imposing constraint 2

in the sub-line for the door assembly it might increase the productivity for the door assembly only, but it does not increase the productivity for the entire oven. Moreover, it increases operator idle time in the door assembly line. Besides, it needs to redesign the existing assembly line. Therefore, only linear sequences are of interest in this case. Another possible reason is that the stacking nature of the product demands linear sequences to assemble it.

By eliminating all the sequences with subassemblies and retaining linear sequences only, the resulting assembly states and tasks are listed in Table 9.9. This constraint reduced the number of states and tasks to 40 and 79 respectively.

At this point, since the complexity has been reduced considerably, the sequences are mapped on the ASG as shown in Figure 9.8. In this figure, the legend describes the significance of each cell. The code letters inscribed in the cells indicate the components. From now on, the editing process will be carried out on the ASG, since the graph gives a better visual representation and one can clearly see which nodes are connected to which arcs.

Constraint 3; Specified order of components: The component air-wash panel (f) must be air washed to remove any dust particles on it before and after assembling it. If the trim frame (h) is assembled before assembling the air-wash panel, air washing of the subassembly is inconvenient. Moreover, since the air-wash panel acts as a housing to the components b,d and e and holds them in place, if it is assembled at the earliest after assembling them.

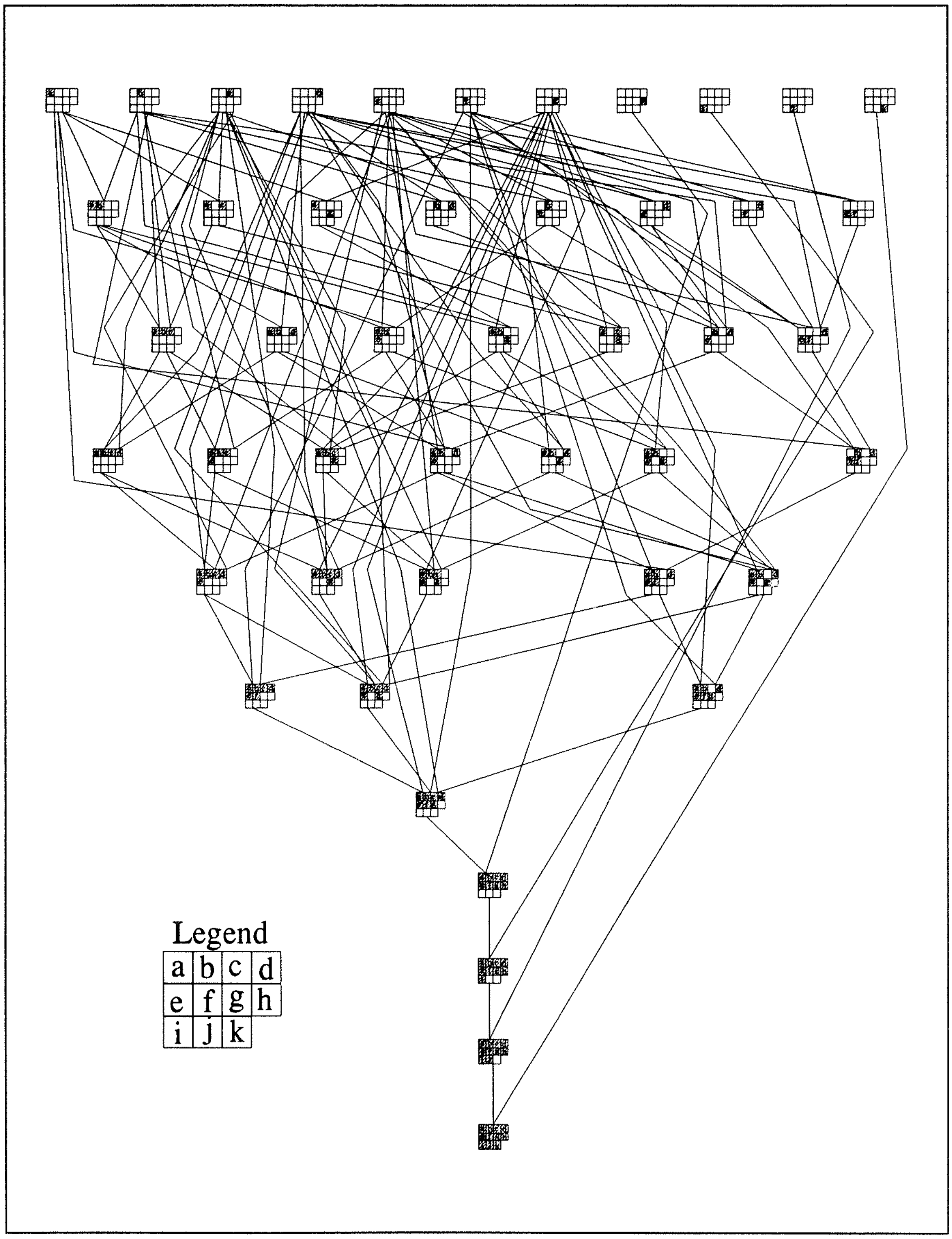


Figure 9.9. The ASG for the oven door assembly after imposing constraint 3

Table 9.10. The AST for the oven door assembly after imposing constraint 3

Level	Nodes (States)	Arcs (Tasks)
2	{a,b}	{(a), (b)}
	{a,c}	{(a), (c)}
	{a,g}	{(a), (g)}
	{b,d}	{(b), (d)}
	{b,e}	{(b), (e)}
	{d,e}	{(d), (e)}
	{d,f}	{(d), (f)}
	{e,f}	{(e), (f)}
3	{a,b,c}	{(a,b), (c)}; {(a,c), (b)}
	{a,b,d}	{(a,b), (d)}
	{a,b,e}	{(a,b), (e)}; {(b,e), (a)}
	{a,b,g}	{(a,b), (g)}; {(a,g), (b)}
	{a,c,g}	{(a,c), (g)}; {(a,g), (c)}
	{b,d,e}	{(b,d), (e)}; {(b,e), (d)}; {(d,e), (b)}
	{d,e,f}	{(d,e), (f)}; {(d,f), (e)}; {(e,f), (d)}
4	{a,b,c,d}	{(a,b,c), (d)}; {(a,b,d), (c)}
	{a,b,c,e}	{(a,b,c), (e)}; {(a,b,e), (c)}
	{a,b,c,g}	{(a,b,c), (g)}; {(a,b,g), (c)}; {(a,c,g), (b)}
	{a,b,d,e}	{(a,b,d), (e)}; {(a,b,e), (d)}; {(b,d,e), (a)}
	{a,b,d,g}	{(a,b,d), (g)}; {(a,b,g), (d)}
	{a,b,e,g}	{(a,b,e), (g)}; {(a,b,g), (e)}
	{b,d,e,f}	{(b,d,e), (f)}; {(d,e,f), (b)}
5	{a,b,c,d,e}	{(a,b,c,d), (e)}; {(a,b,c,e), (d)}; {(a,b,d,e), (c)}
	{a,b,c,d,g}	{(a,b,c,d), (g)}; {(a,b,c,g), (d)}; {(a,b,d,g), (c)}
	{a,b,c,e,g}	{(a,b,c,e), (g)}; {(a,b,c,g), (e)}; {(a,b,e,g), (c)}
	{a,b,d,e,f}	{(a,b,d,e), (f)}; {(b,d,e,f), (a)}
	{a,b,d,e,g}	{(a,b,d,e), (g)}; {(a,b,d,g), (e)}; {(a,b,e,g), (d)}

Table 9.10. (Continuation)

Level	Nodes (States)	Arcs (Tasks)
6	{a,b,c,d,e,f}	{(a,b,c,d,e), (f)}; {(a,b,d,e,f), (c)}
	{a,b,c,d,e,g}	{(a,b,c,d,e), (g)}; {(a,b,c,d,g), (e)}; {(a,b,c,e,g),(d)} {(a,b,d,e,g), (c)}
	{a,b,d,e,f,g}	{(a,b,d,e,f), (g)}; {(a,b,d,e,g), (f)}
7	{a,b,c,d,e,f,g}	{(a,b,c,d,e,f), (g)}; {(a,b,c,d,e,g), (f)} {(a,b,d,e,f,g), (c)}
8	{a,b,c,d,e,f,g,h}	{(a,b,c,d,e,f,g), (h)}
9	{a,b,c,d,e,f,g,h,i}	{(a,b,c,d,e,f,g,h), (i)}
10	{a,b,c,d,e,f,g,h,i,j}	{(a,b,c,d,e,f,g,h,i), (j)}
11	{a,b,c,d,e,f,g,h,i,j,k}	{(a,b,c,d,e,f,g,h,i,j), (k)}

Total number of states = 35   Total number of tasks = 68

According to above discussion, any assembly state that contains the component h without having the component f assembled is undesirable. The process planner has to locate the nodes in which the cell representing the component h is filled. If the cell f is not filled in these nodes, they are not desirable and hence they must be deleted. From Figure 9.8 and Table 9.9, it can be observed that the assembly states {a,b,c,d,e,g,h}, {a,b,c,e,g,h}, {a,b,c,d,g,h}, {a,b,c,g,h} and {a,c,g,h} are not desirable. After all deleting all these nodes and the related hyperarcs, the AST and ASG for the remaining nodes and arcs are presented in the Table 9.10 and Figure 9.9 respectively.

Constraint 4; Subassembly stability (phase 1): Though the spacer (g) can be added to door body (a) at any time before assembling the trim frame (h), it is preferable to assemble it just before assembling the trim frame. It is because spacer is not attached securely to the door body and may be disturbed during subsequent assembly operations. Since the trim frame encompasses the spacer, it retains the spacer in its position.

Table 9.11. The AST for the oven door assembly after imposing constraint 4

Level	Nodes (States)	Arcs (Tasks)
2	{a,b}	{(a), (b)}
	{a,c}	{(a), (c)}
	{b,d}	{(b), (d)}
	{b,e}	{(b), (e)}
	{d,e}	{(d), (e)}
	{d,f}	{(d), (f)}
	{e,f}	{(e), (f)}
3	{a,b,c}	{(a,b), (c)}; {(a,c), (b)}
	{a,b,d}	{(a,b), (d)}
	{a,b,e}	{(a,b), (e)}; {(b,e), (a)}
	{b,d,e}	{(b,d), (e)}; {(b,e), (d)}; {(d,e), (b)}
	{d,e,f}	{(d,e), (f)}; {(d,f), (e)}; {(e,f), (d)}
4	{a,b,c,d}	{(a,b,c), (d)}; {(a,b,d), (c)}
	{a,b,c,e}	{(a,b,c), (e)}; {(a,b,e), (c)}
	{a,b,d,e}	{(a,b,d), (e)}; {(a,b,e), (d)}; {(b,d,e), (a)}
	{b,d,e,f}	{(b,d,e), (f)}; {(d,e,f), (b)}
5	{a,b,c,d,e}	{(a,b,c,d), (e)}; {(a,b,c,e), (d)}; {(a,b,d,e), (c)}
	{a,b,d,e,f}	{(a,b,d,e), (f)}; {(b,d,e,f), (a)}
6	{a,b,c,d,e,f}	{(a,b,c,d,e), (f)}; {(a,b,d,e,f), (c)}
7	{a,b,c,d,e,f,g}	{(a,b,c,d,e,f), (g)}
8	{a,b,c,d,e,f,g,h}	{(a,b,c,d,e,f,g), (h)}
9	{a,b,c,d,e,f,g,h,i}	{(a,b,c,d,e,f,g,h), (i)}
10	{a,b,c,d,e,f,g,h,i,j}	{(a,b,c,d,e,f,g,h,i), (j)}
11	{a,b,c,d,e,f,g,h,i,j,k}	{(a,b,c,d,e,f,g,h,i,j), (k)}

Total number of states = 24 Total number of tasks = 39

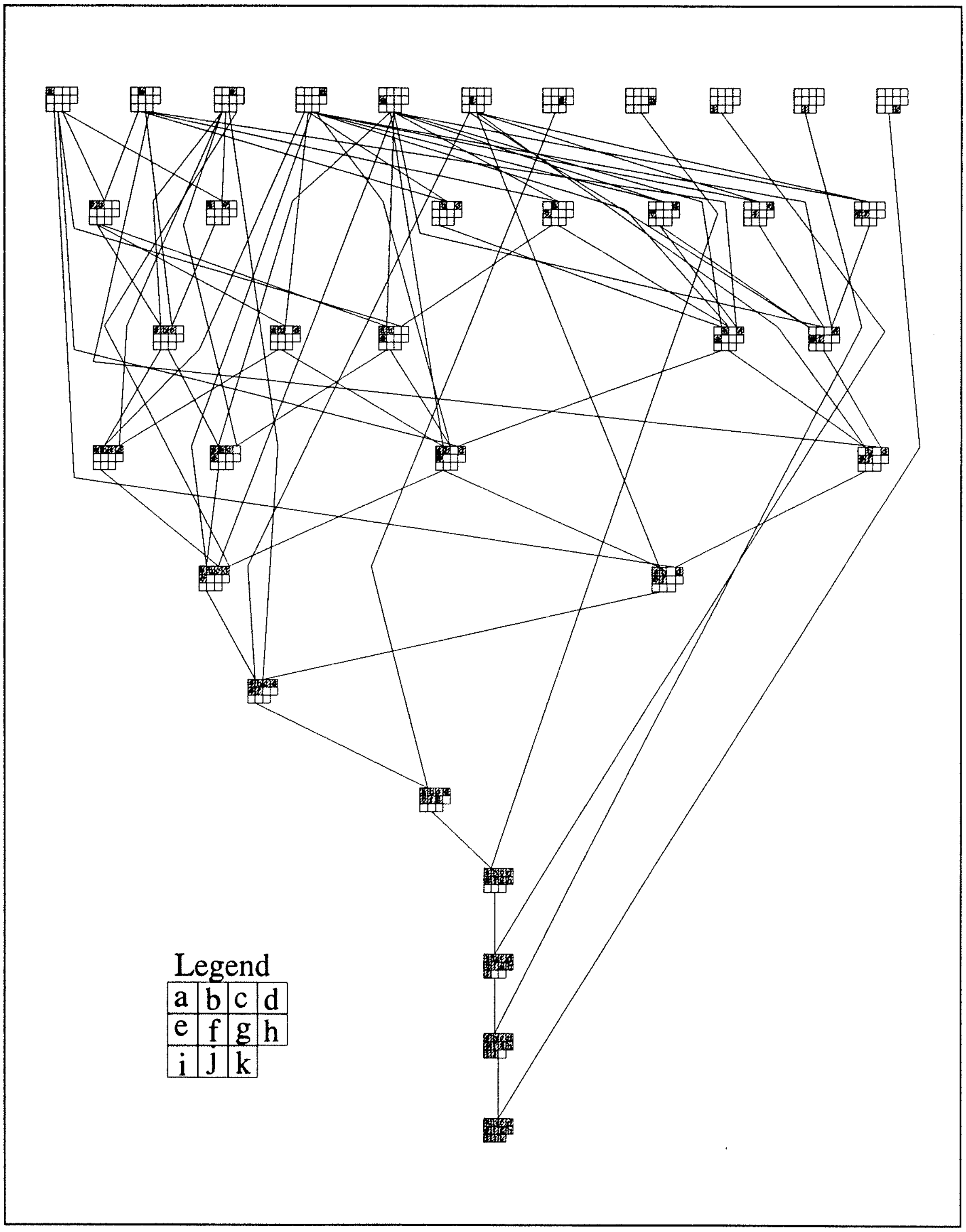


Figure 9.10. The ASG for the oven door assembly after imposing constraint 4

From the Table 9.10 and Figure 9.9, it can be seen that the component h is assembled at level 8. Hence, any assembly task or assembly state that involves the component g till level 6 is not desirable. Elimination of all these states and related tasks will delete the assembly tasks {(a,b,c,d,e,g), (f)} and {(a,b,d,e,f,g), (c)} in the level 7. This constraint reduces the number of assembly staes and tasks to 24 and 39 respectively as shown in Figure 9.10 and Table 9.11.

Constraint 5; Subassembly stability (phase 2): The subassemblies {b,d,e,f}, {b,d,e}

Table 9.12: The AST for the oven door assembly after imposing constraint 5

Level	Nodes (States)	Arcs (Tasks)
2	{a,b}	{(a), (b)}
	{a,c}	{(a), (c)}
3	{a,b,c}	{(a,b), (c)}; {(a,c), (b)}
	{a,b,d}	{(a,b), (d)}
	{a,b,e}	{(a,b), (e)};
4	{a,b,c,d}	{(a,b,c), (d)}; {(a,b,d), (c)}
	{a,b,c,e}	{(a,b,c), (e)}; {(a,b,e), (c)}
	{a,b,d,e}	{(a,b,d), (e)}; {(a,b,e), (d)}
5	{a,b,c,d,e}	{(a,b,c,d), (e)}; {(a,b,c,e), (d)}; {(a,b,d,e), (c)}
	{a,b,d,e,f}	{(a,b,d,e), (f)}
6	{a,b,c,d,e,f}	{(a,b,c,d,e), (f)}; {(a,b,d,e,f), (c)}
7	{a,b,c,d,e,f,g}	{(a,b,c,d,e,f), (g)}
8	{a,b,c,d,e,f,g,h}	{(a,b,c,d,e,f,g), (h)}
9	{a,b,c,d,e,f,g,h,i}	{(a,b,c,d,e,f,g,h), (i)}
10	{a,b,c,d,e,f,g,h,i,j}	{(a,b,c,d,e,f,g,h,i), (j)}
11	{a,b,c,d,e,f,g,h,i,j,k}	{(a,b,c,d,e,f,g,h,i,j), (k)}

Total number of states = 16 Total number of tasks = 23

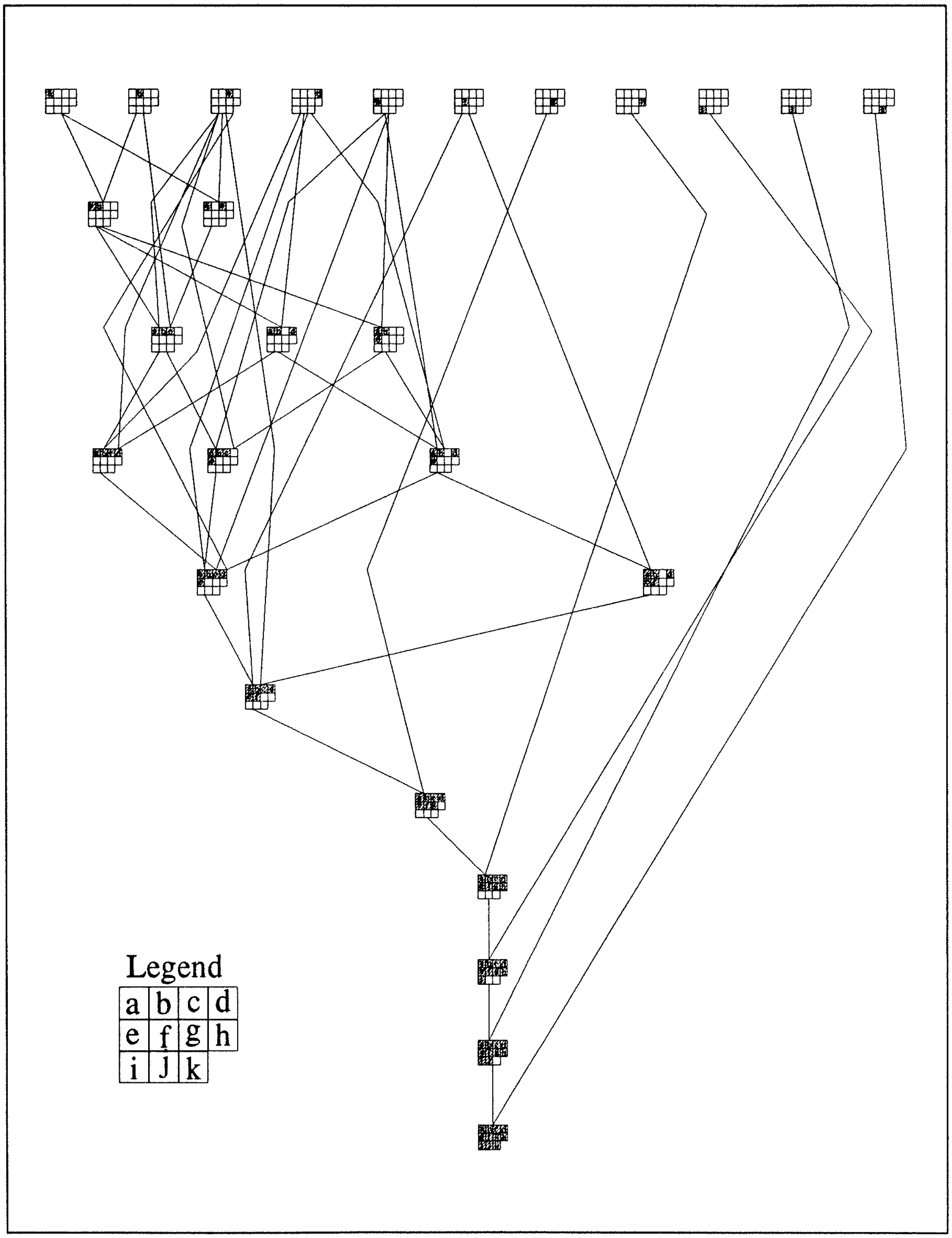


Figure 9.11. The AST for the oven door assembly after imposing constraint 5

and {b,e} do not have any attachment agents and they are not stable subassemblies. As discussed earlier, after attaching the component f to a, it covers the components b,d and e, and then acts as a stable subassembly. Therefore, the assembly states {b,d,e,f}, {b,d,e} and {b,e} are not desirable.

Table 9.12 gives the list of assembly states and tasks after deleting the above states and related tasks. The graphical representation is shown in Figure 9.11. With this constraint, the numbers assembly states and assembly tasks have been reduced to 17 and 25 respectively.

Constraint 6; Difficult to do assembly tasks: Some assembly tasks, though they are feasible, it is difficult to maintain the position components of the involved subassemblies during the assembly task. They require extra fixtures or holding agents to keep them in position during the assembly task. In this example, handling the subassemblies {a,b}, {a,b,d}, {a,b,e}, {a,b,d,e} require some extra holding devices during the assembly of component c. Assembling component c to a before adding other components is preferable, since c is attached securely to a and it does not require any holding devices during the subsequent assembly operations. After deleting all the assembly tasks involving the above mentioned subassemblies and the component c, the remaining sequence possibilities are presented in Table 9.13 and Figure 9.12.

9.3.4 Sequence Evaluation

The consideration of several selection criteria has allowed us to eliminate a number of feasible sequences and we are left with four assembly sequences which are represented in the ASG shown in Figure 9.12. These sequences must be

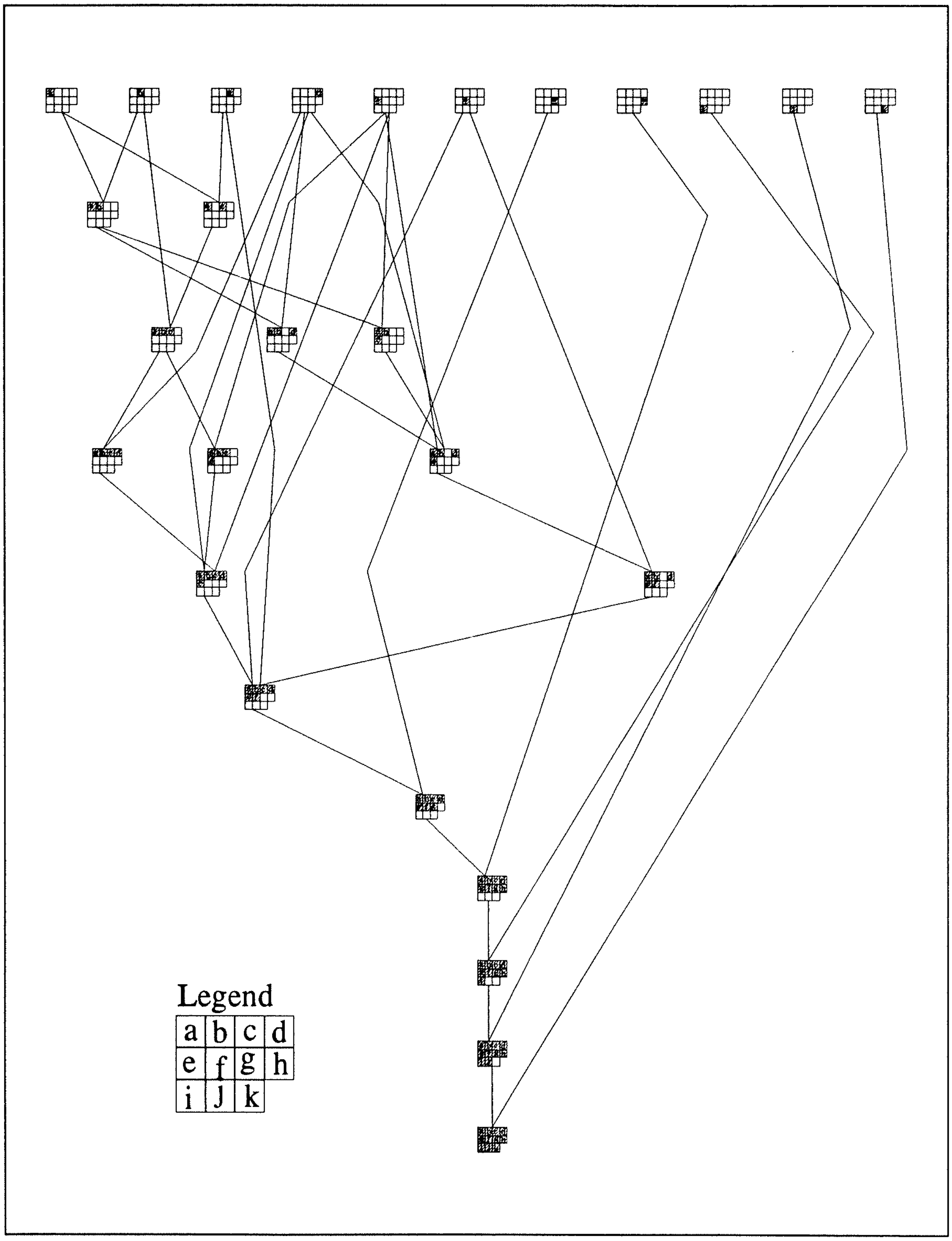


Figure 9.12. The ASG for the oven door assembly after imposing constraint 6

Table 9.13. The AST for the oven door assembly after imposing constraint 6

Level	Nodes (States)	Arcs (Tasks)
2	{a,b}	{(a), (b)}
	{a,c}	{(a), (c)}
3	{a,b,c}	{(a,c), (b)}
	{a,b,d}	{(a,b), (d)}
	{a,b,e}	{(a,b), (e)}
4	{a,b,c,d}	{(a,b,c), (d)}
	{a,b,c,e}	{(a,b,c), (e)}
	{a,b,d,e}	{(a,b,d), (e)}; {(a,b,e), (d)}
5	{a,b,c,d,e}	{(a,b,c,d), (e)}; {(a,b,c,e), (d)}
	{a,b,d,e,f}	{(a,b,d,e), (f)}
6	{a,b,c,d,e,f}	{(a,b,c,d,e), (f)}; {(a,b,d,e,f), (c)}
7	{a,b,c,d,e,f,g}	{(a,b,c,d,e,f), (g)}
8	{a,b,c,d,e,f,g,h}	{(a,b,c,d,e,f,g), (h)}
9	{a,b,c,d,e,f,g,h,i}	{(a,b,c,d,e,f,g,h), (i)}
10	{a,b,c,d,e,f,g,h,i,j}	{(a,b,c,d,e,f,g,h,i), (j)}
11	{a,b,c,d,e,f,g,h,i,j,k}	{(a,b,c,d,e,f,g,h,i,j), (k)}

Total number of states = 16   Total number of tasks = 19

carefully evaluated to select the final sequence. Because of the stacking nature of the product, all the selected sequences have almost the same assembly tasks except with a difference in one or two assembly tasks. The cost of each assembly task indicated in Figure 9.12 has been estimated using the DFA Toolkit software and these costs are assigned to the corresponding arcs. The ASG shown in Figure 9.13 includes the assembly costs in cents resulted from the DFA software. This cost analysis has been performed based on a total hourly labor rate of $30.00 including basic pay and all fringe benefits. It should be noted that while estimating these costs, the software does not take the efficiency of operator or other equipment cost into account.

If any component uses a fixture, the cost of placing that component in the fixture is assigned to the corresponding node in the first level. For example, in the Figure 9.13, the nodes representing the components a and c are associated with the fixturing operation costs.

In the example under study, it is convenient to place the door hinges on a specially designed fixture and the door body (or the door body fitted with other components) will be attached to the hinges by means of a screwing operation. Then this subassembly is reoriented, so that other components can be added to it. These reorientations or any other special operations such as cleaning, inspection, testing, etc. are considered as separate assembly operations in DFA. The costs of these additional operations to assemble a component are included in the assembly cost of that component by assigning the appropriate cost factors to the arcs in the ASG. For example, the cost of assembling the air-wash panel includes the cost of air cleaning the component before its assembly and air cleaning the subassembly after its assembly. The cost of each air cleaning operation is estimated by DFA software as 4 cents. Since air cleaning is done before and after the assembly of the air-wash panel, total air cleaning cost is 8 cents. Cost of assembling air wash panel is estimated at 13 cents. Hence, a total cost of 21 cents is assigned to the arc that corresponds to the assembly of component f in the ASG. Similarly, the cost of other assembly tasks assigned to the corresponding arcs in the ASG includes the cost of other additional operations required to that assembly task.

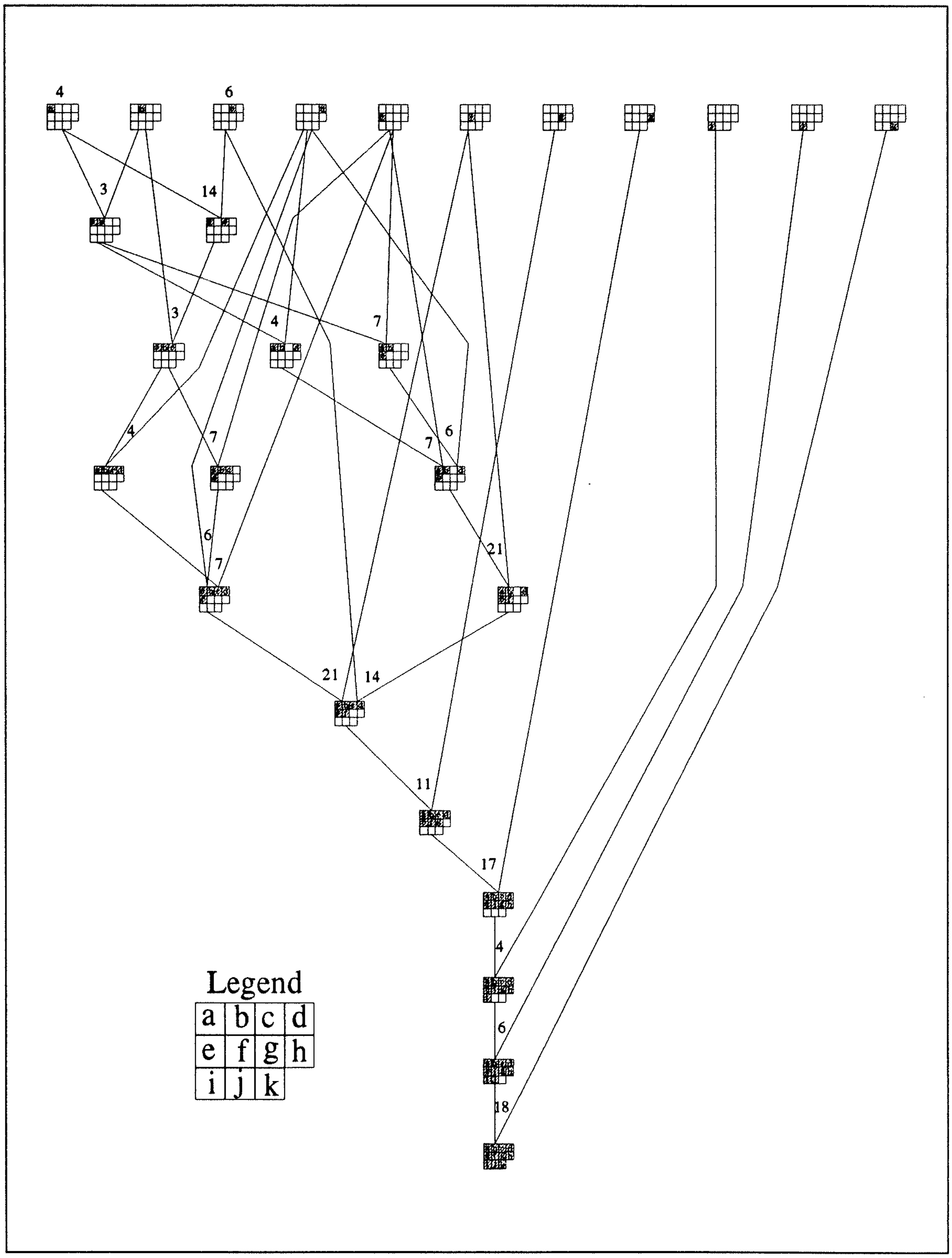


Figure 9.13. The ASG for the oven door assembly with the associated cost factors

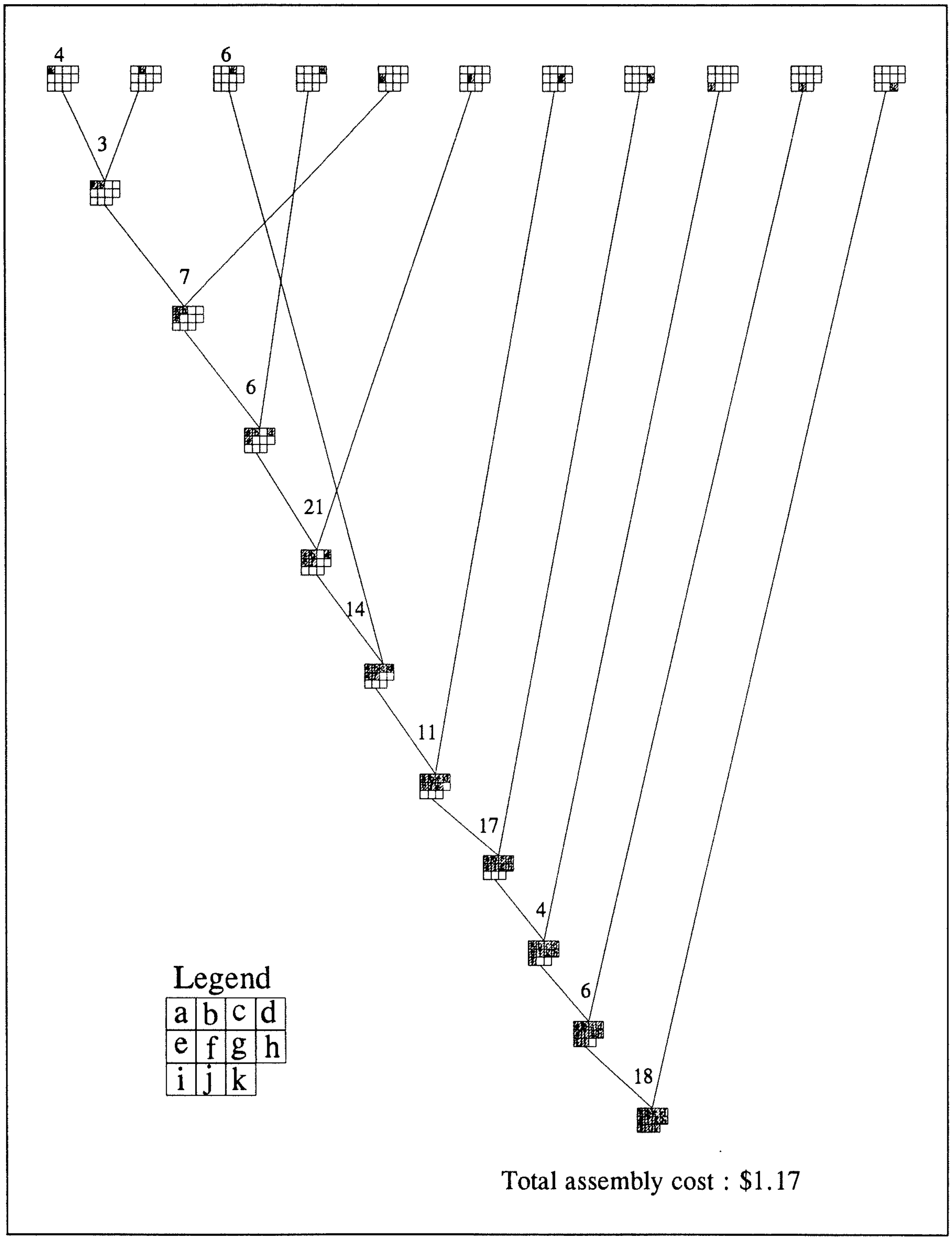


Figure 9.14a. The selected assembly sequences for the oven door assembly (sequence 1)

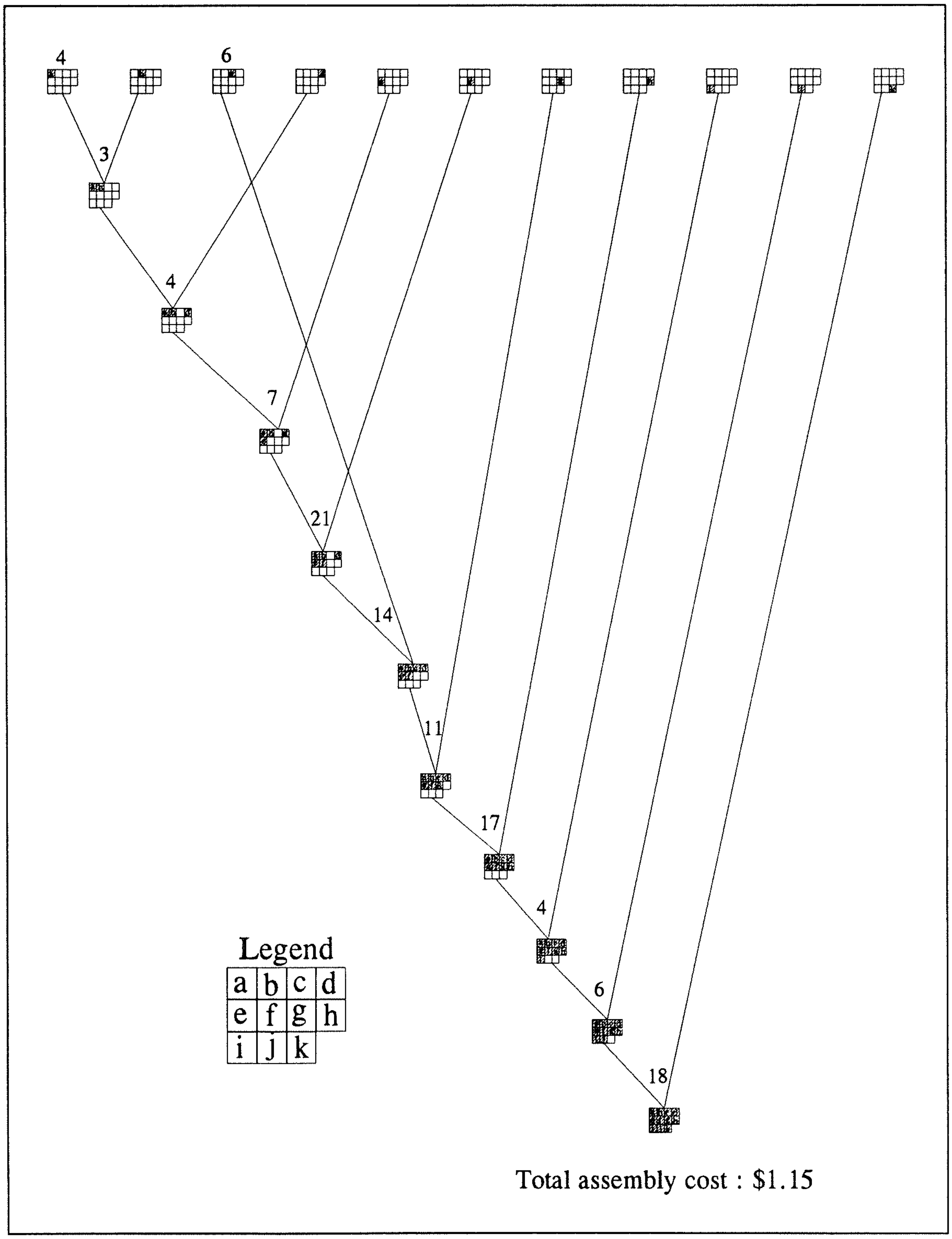


Figure 9.14b. The selected assembly sequences for the oven door assembly (sequence 2)

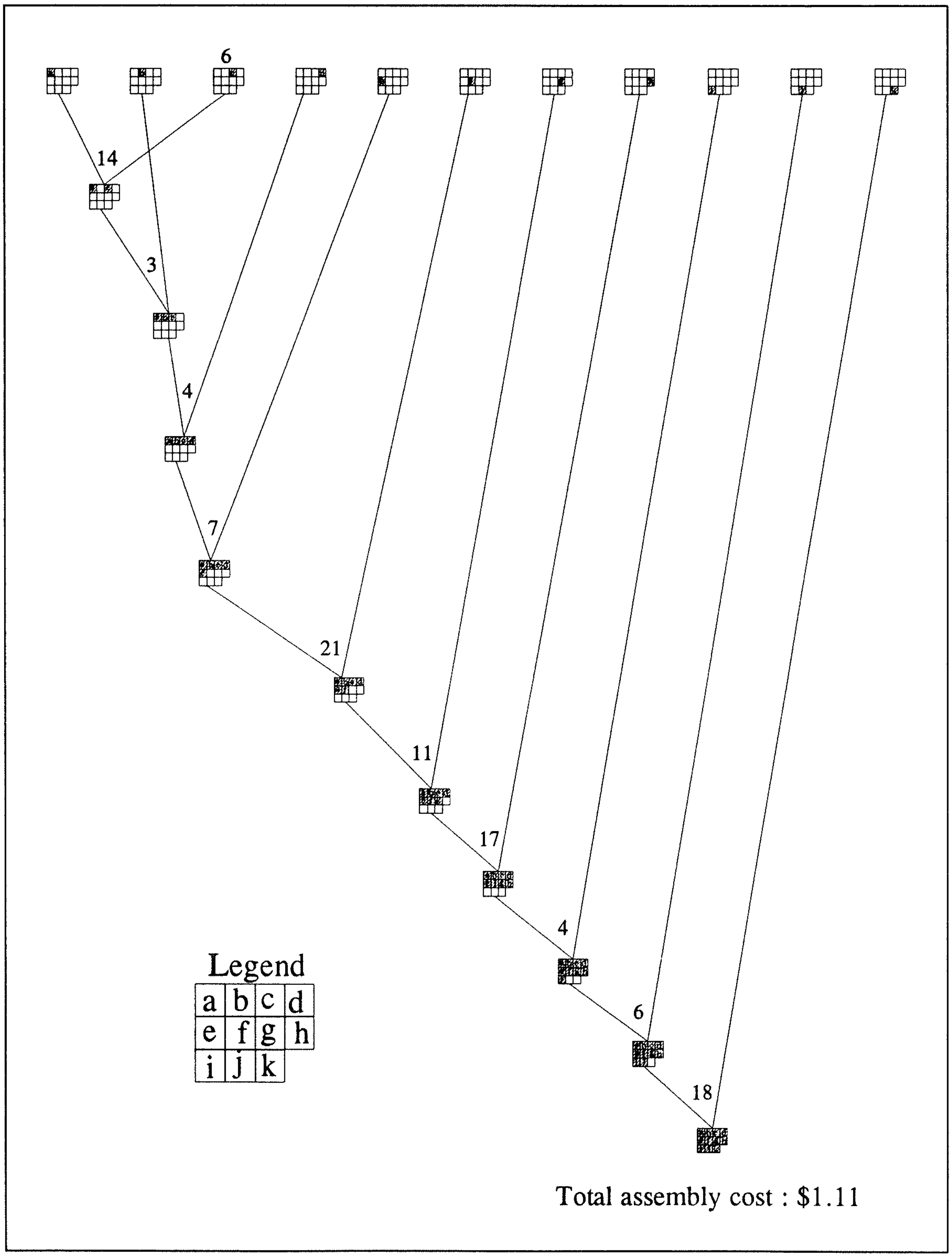


Figure 9.14c. The selected assembly sequences for the oven door assembly (sequence 3)

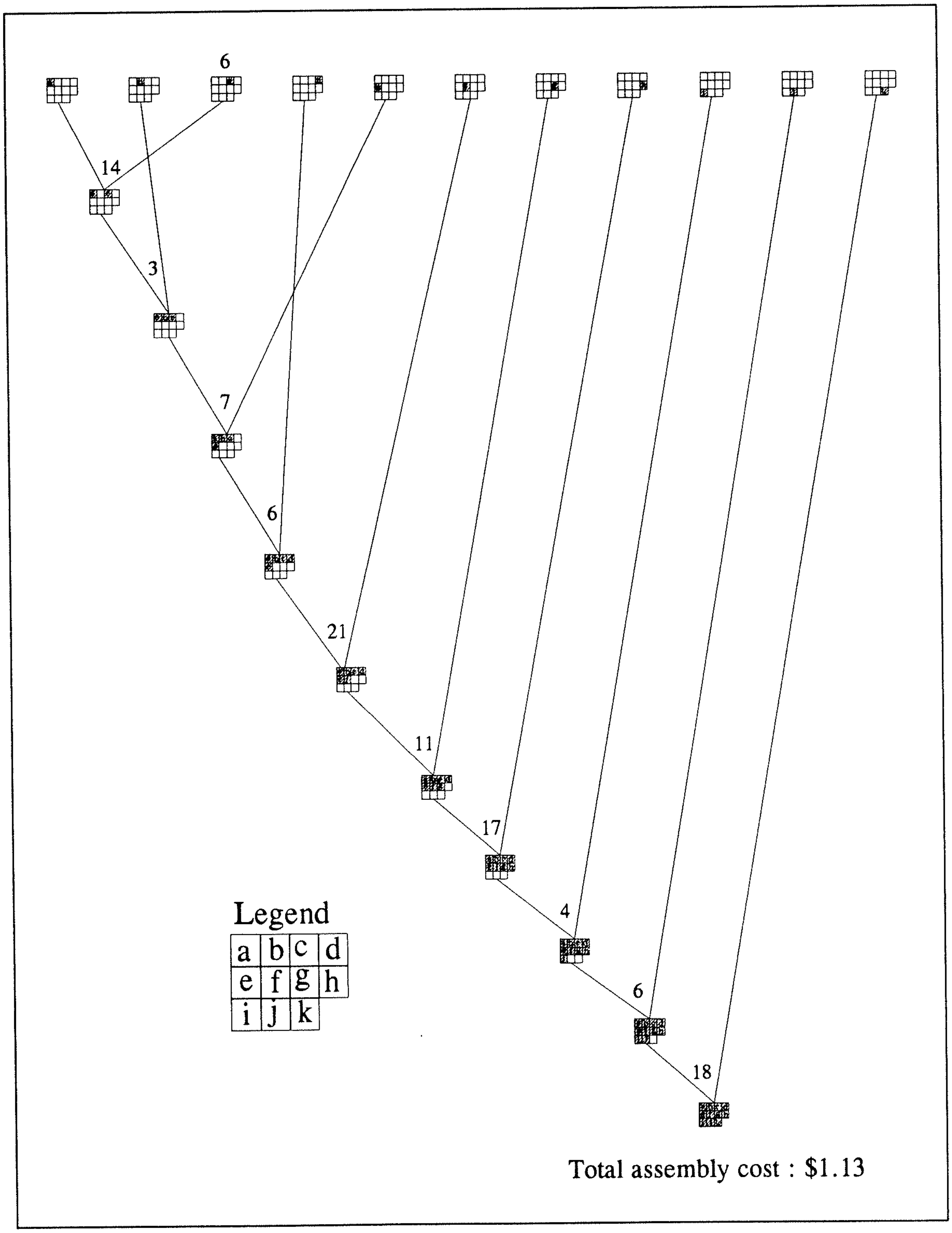


Figure 9.14d. The selected assembly sequences for the oven door assembly (sequence 4)

Figure 9.14 shows all the individual sequences that can be derived from the ASG shown in Figure 9.13, with their associated assembly task costs. The sum of all the costs in a particular sequence is the total assembly cost if the product is assembled in that sequence. It can be observed that the total assembly cost for the sequence 3 presented in Figure 9.14c is $1.11, which is the smallest compared to the costs of other three sequences. Hence, sequence 3 is selected as the best choice to assemble the given product. It is interesting to note that the same sequence is being used in the Frigidaire plant to assemble the oven door. This sequence had been selected at Frigidaire by a trial-and-error approach.

9.3.5 Effect of Selection and Evaluation Criteria

The various constraints used in the selection and evaluation process, and their effect on the number of possible assembly states and tasks is summarized in Table 14. The same result is plotted on a graph shown in Figure 9.15. It resembles the "law of diminishing returns", i.e., the effect of each constraint in reducing the number of possible sequences gradually become less important. After a certain limit, imposition of extra constraints will not have any effect in reducing the sequence count. Figure 9.15 clearly depicts that Constraint 3 does not have much effect. It should also be noted that, if the same constraint would have been applied earlier, its effect would have been different. Though the effect of individual constraints depends on the order they are applied, the net result would be the same irrespective of their order. That means, even if the order of the six selection constraints that were used here is changed, there will not be any change in the final result. Since the application of last constraint, i.e., assembly cost, has resulted only one sequence, the number of states and tasks are the same (which is equal to 10).

Table 9.14. Effect of selection and evaluation criteria on the number of assembly states and tasks

No.	Constraint	Number of Assembly States	Number of Assembly Tasks
0	No constraint	71	249
1	Specified last component	46	141
2	Linearity	40	79
3	Specified order of components	34	69
4	Subassembly stability (Phase I)	24	39
5	Subassembly stability (Phase II)	16	23
6	Difficult to handle assembly tasks	16	19
7	Assembly cost	10	10

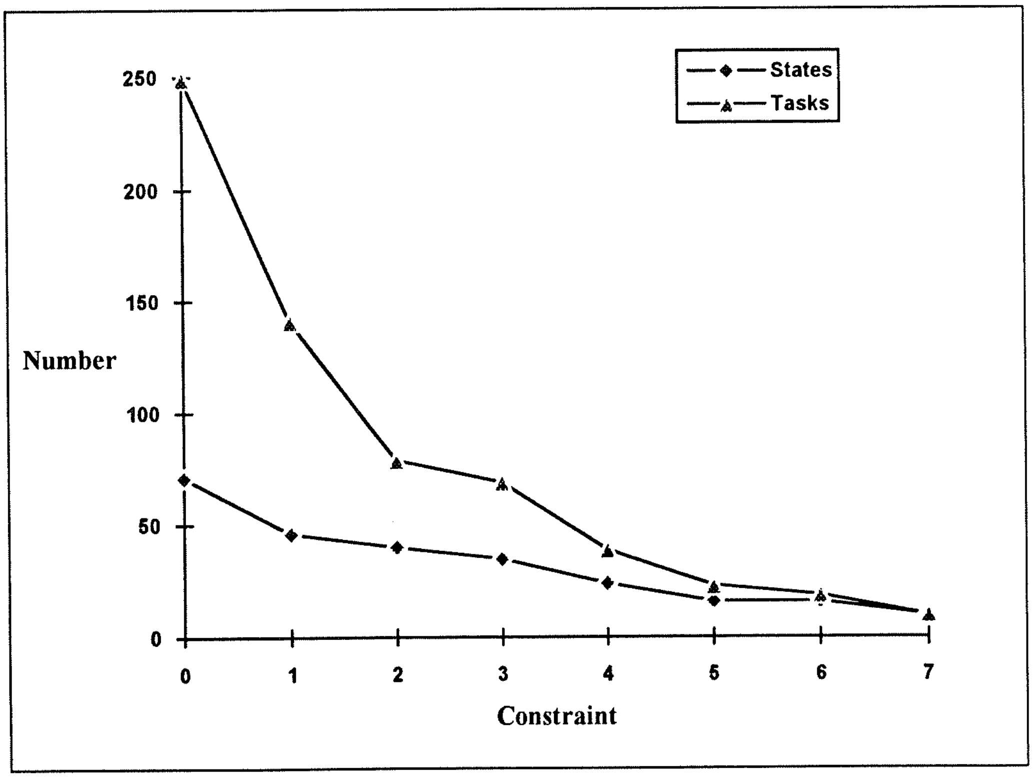


Figure 9.15. Effect of selection and evaluation criteria on the number of sequences

The sequence reduction process must be stopped, whenever the curves representing the number of states and the number of tasks meet, which indicates that there is only one assembly sequence left.

9.3.6 Design Modifications

The fundamental principle of "Design for assembly" is to reduce the number of components that must be assembled. In the previous discussion, the cost of assembly of two spacers in the oven door assembly is estimated as 11 cents. The function of the spacer is merely to provide some support between the body and the trim frame. The same function can be fulfilled by a slight modification in the design of the door body as shown in Figure 9.16.

In the new design, two welded supports are used in order to replace the spacers. Obviously, it increases the production cost of the door body. But, it totally eliminates the production cost or the purchasing cost of the spacers, which will result in a substantial reduction of the total cost of the product. Moreover, the door body already has four welded supports to which the air wash panel is securely attached. Adding two more metal supports in the same welding setup does not increase the cost that much.

The new design completely eliminates the use of spacers, which shows savings in the assembly cost by 11 cents. In addition, with the elimination of the spacers, the assembly of screws does not require any extra alignment, which reduces their assembly cost from 18 cents to 13 cents. Thus, there will be a reduction of 16 cents in the total assembly cost, which is about 14.4% of the total assembly cost before redesign. Moreover, complete elimination of a component shows substantial savings in the inventory cost and administration cost as well.

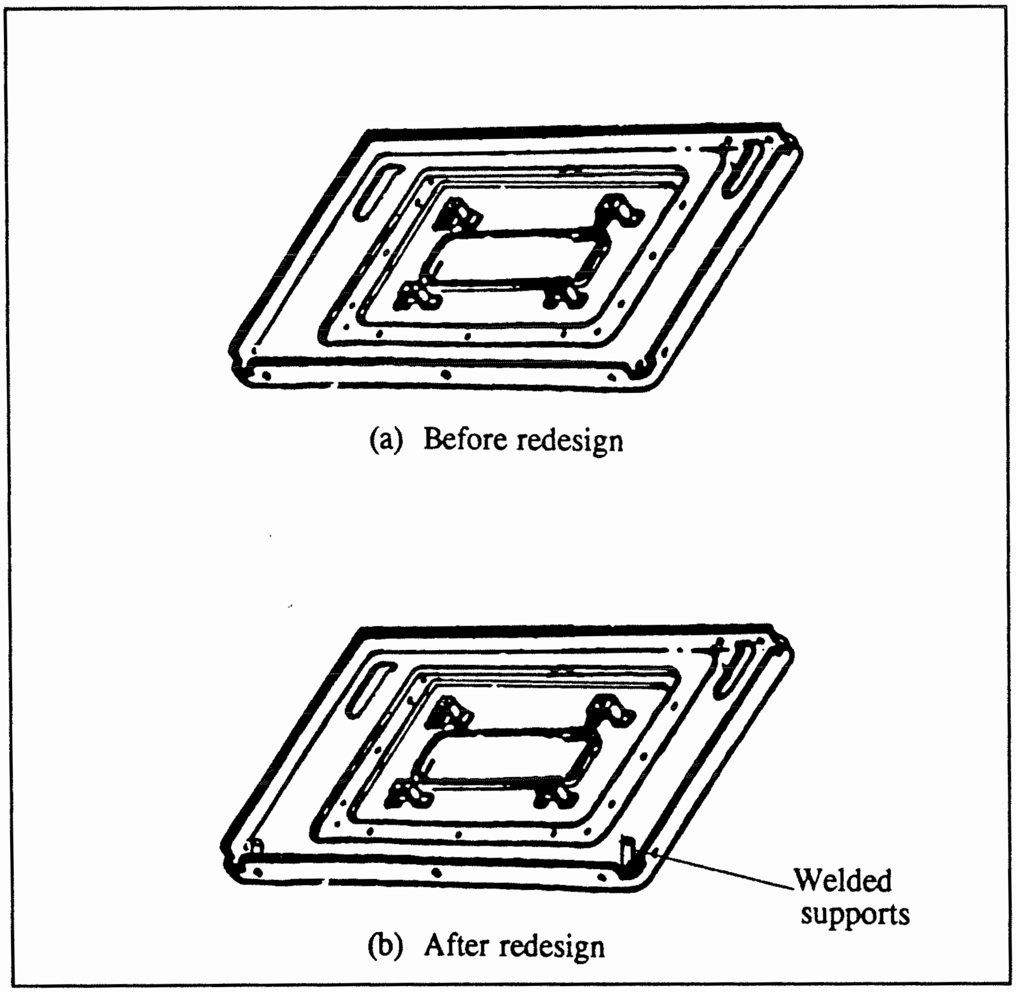


Figure 9.16. Redesign of door body to eliminate spacers

The reduction of the number of components of an assembly not only reduces the assembly cost, but also reduces the complexity of the assembly planning procedure. With the new design, the door assembly has only 10 components. The total number of pairs for which C and T functions are to be generated and analyzed will be reduced to 45 from 55. The number of probable combinations to be analyzed for assembly feasibility, and the number of possible assembly states and tasks are much smaller. The ASG will have only 10 levels, and each node or box will consist of only 10 cells instead of 11.

The examples of the ball point pen and the oven door assembly demonstrate the applicability and usefulness of the proposed approach in solving the assembly planning problems for real industrial products. The described method clearly exhibits some superiority over the previous assembly sequence determination methods. The implementation of "Design for assembly" to modify the design not only reduces the assembly cost, but also reduces the complexity of the assembly sequence generation procedure.

CONCLUSION

A computer integrated assembly planning system (CIAPS) capable of systematically determining the best assembly sequence from the CAD description of a product has been presented. The entire assembly sequence planning has been carried out in three phases. In the first phase, the CSG representation of the product to be assembled is created using the PADL-2 solid modeling system. The information concerning the connectivity relations and collision constraints has been automatically extracted from the geometric model of the assembly. This geometric reasoning has been performed by employing the collision detection procedure for each pair of components of the assembly. One of the most important features of the proposed approach is its ability to transform a three-dimensional description of the assembly into connectivity and motion constraints. The extracted information is stored in the form of a relational model. The representation of precedence constraints in terms of contact and translational functions in the relational model has a number of advantages: it is binary in nature, ant it is easy to visualize and interpret. Any new information can be included in the relational model, thus making the system very flexible.

In the next phase, the assembly sequence generation algorithm is used to transform the C and T binary functions into a set of feasible assembly sequences. The binary nature of relational information enabled the sequence generation algorithm to use simple Boolean operations to generate the geometrically feasible sequences. The two different schemes, assembly sequence graph and assembly sequence table that are used to represent the generated assembly schemes provide a convenient way for further analysis of sequences. These two schemes are complete as they encompass all the feasible sequences.

In the last phase, the generated sequences are winnowed by imposing non-geometric and strategic constraints. The editing features of the ASG or AST described in Chapter 6 to eliminate unwanted assembly sequences (according to the criteria defined by the user) has made the method flexible and efficient. The assembly sequence selection criteria described in this research significantly influences the unit assembly cost and assembly system configuration. The selected sequences are then subjected to cost analysis using the DFA Toolkit software. The final sequence is feasible and optimal in terms of assembly time and assembly cost. This phase of considering the constraints imposed by non-geometric properties associated with the design of the product as well as the assembly facility makes the final plans practical enough to be used in real life situations. The sequence representation schemes are well structured and suited to implement several qualitative and quantitative criteria.

The computational complexity of the sequence generation algorithm in generating all geometrically feasible sequences was assessed by determining the number of probable combinations that must be analyzed. This amount depends not only on the number of parts, but also on how they are interconnected. The measures for this complexity were estimated in terms of number of assembly states and assembly tasks that can be obtained.

The proposed integrated assembly planning system CIAPS provides a systematic method to determine the assembly sequence from a 3-D solid modeler description of an assembly. This system first considers geometric constraints and then non-geometric constraints to ensure the generated assembly plans are feasible and realistic. In addition to determining the assembly sequences, the system is also capable of identifying the assembly direction of the component.

The method detects obstructions for every candidate component trajectory by performing collision detection with respect to the other components. This results in a complete set of connectivity and precedence constraints. It can be observed that the majority of the execution time is spent on reasoning with the assembly geometric model, especially on performing collision detection among the component parts of the assembly. Once the part mating and collision information has been derived from the assembly geometry, the assembly plans are generated in a short time. In spite of the complexity of the collision detection, real product assemblies can be handled by the system effectively.

The illustrated examples clearly demonstrated the practical applicability of the system in solving the assembly planning problems. An important point here is the selected final assembly sequence for the oven-door assembly exactly coincides with the sequence that is being currently used to assemble the product at Frigidaire, which they have developed after several experimental trials. It clearly indicates the efficiency of this method in determining the assembly sequence in the early design phase, without making any trials.

Another important salient feature of this method is its ability to extend for the assemblies with higher parts counts. It does not require a tedious question answer session to generate the precedence knowledge, that is essential in many of the methods that are currently available. Instead, it has been done automatically from the CAD model.

The prposed user interface allows the user to intervene in making the decisions that are otherwise difficult. Also, the design modification provision facilitates the implementation of the "Design for assembly" principles. In summary, CIAPS will become a useful aid that assists the product designer and the manufacturing engineer to cope with today's rapidly changing product designs and manufacturing facilities and to implement concurrent engineering concepts.

RECOMMENDATIONS

Despite the described advantages of the proposed approach, there are some other aspects that must be addressed to improve its efficiency and applicability and to put the automatic assembly planning into practice.

The level of integration between the CAD system and the DFA software should be increased, so that all the geometric and non-geometric features of the parts can be extracted and translated into handling and insertion codes.

The level of automation can be further improved by including the non-geometric characteristics of the component parts in the relational model of the assembly. This information can be used during the selection and evaluation phase with minimal human intervention. The inclusion of other strategic constraints such as resource constraints, facility constraints, etc., in the relational model would further enhance the capabilities of the system to come to the final decision automatically.

It is suggested to enhance the system to perform the contact analysis using the C, T and R functions. It is useful in the assembly process design.

It is also suggested to make provision to consider the tolerances on the dimensions of components. For the variations in the dimensions of individual components, the position of other components must be adjusted and the precedence knowledge must be updated accordingly.

At present, the system does not include any modules for assembly operation planning and off-line programming of assembly machines. If the system is updated to include this, it will become a full-fledged automatic assembly planning system that incorporates all the levels of planning from the design to the assembly operation.

REFERENCES

1. BALDWIN, D.F., ABELL, T.E., LUI, M.C., DE FAZIO, T.L. and WHITNEY, D.E., 1991 "An integrated computer aid for generating and evaluating assembly sequences for mechanical products", IEEE Transactions on Robotics and Automation, 7(1), 78-94.

2. BARAKAT, O. and VALLET, G., 1989 "Choice of assembly sequences by simulation and data analysis", Proceedings of the 17th ASTED Conference on Simulation and Modelling, Lugano, Switzerland.

3. BEN-ARIEH, D and KRAMER, B., 1994 "Computer-aided process planning for assembly: generation of assembly operation sequences", International Journal of Production Research, 32(3), 643-656.

4. BONSCHANCHER, N. and HEEMSKERK, C.J.M., 1989 "Grouping parts to reduce the complexity of assembly sequence planning", Proceedings of IFAC International Conference on Control Problems in Manufacturing Technology, Madrid, Spain, pp. 233-238.

5. BOOTHROYD, G., 1991 "Assembly automation and product design", Marcel Dekker, Inc., New York, pp. 413.

6. BOOTHROYD, G. and DEWHURST, P., 1989 "Product design for assembly", Boothroyd Dewhurst, Wakefield, Rhode Island, USA.

7. BOURJAULT, A., 1984 "Contribution à une approche méthodologique de l'assemblage automatisé: élaboration automatique des séquences opératoires", Thèse d'État, Université de Franche-Comté, Besançon, France.

8. BROWN, C.M., 1982 "PADL-2: A technical summary", IEEE Computer Graphics & Applications, 2(2), 69-84.

9. CHANG, K.H. and WEE, W.G. 1988, "A knowledge-based planning system for mechanical assembly using robots", IEEE Expert, 3(1), 18-30.

10. CHANG, P.T., BEN-ARIEH, D. and LEE, E.S., 1992 "Generation of mechanical assembly sequences with fuzzy weights", Proceedings of 1st Industrial Engineering Research Conference, pp. 97-101.

11. CHANG, T.C. and TERWILLINGER, J.P., 1987 "Rule based approach for

printed wiring assembly process planning", Proceedings of 8th International Conference on Assembly Automation, pp. 99-110.

12. CHAY, D., LENS, E. and SHPITALNI, M., 1988 "Picking parameters for easy automation of assembly", Assembly Automation, 8, 151-154.

13. CHEN, C.L.P., 1991 "Automatic assembly sequences generation by pattern matching", IEEE Trans. on Systems, Man and Cybernetics, 21(2), 376-389.

14. CHEN, C.L.P. and WICHMAN, C.A., 1993 "A systematic approach for design and planning of mechanical assemblies", Artificial Intelligence in Engineering Design, Analysis and Manufacturing, 7(1), 19-36.

15. DALLY, T.M., 1985 "BOXER: A design to build system, System architecture study", Technical Report RC 11096 (No. 49852), IBM Thomas Watson Research Center, White Plains, NY, USA.

16. DE FAZIO, T.L. and WHITNEY, D.E., 1987 "Simplified generation of all mechanical assembly sequences", IEEE Journal of Robotics and Automation, RA-3(6), 640-658.

17. DELCHAMBRE, A., 1990 "A pragmatic approach to computer aided assembly planning", Proceedings of IEEE International Conference on Robotics and Automation, Las Alamitos, California, USA, pp. 1600-1605.

18. GELEYN, J.E.D., 1988 "Solid Modelling applications for robotic assembly simulation", Master's Thesis, Royal Military College of Canada, Kingston, Ontario, Canada.

19. GHOSH, K. CARACCIOLO, A. and CWYCYSHYN, W., 1987 "Robotic applications in the assembly of automotive engines", Proceedings of the 17th Intl. Symposium Industrial Robots, Chicago, Illinois, USA, pp. 7.1-7.13.

20. GHOSH, K., 1989, "Extraction des données et systèmes experts pour élaborer les gammes admissables d'assemblage", Rapport technique, CDT Project C095, École Polytechnique, Montréal, Québec, Canada.

21. GHOSH, K. and REDDY, G.B., 1991 "Methods and algorithms for the generation of assembly sequences", Proc. of the 6th International Conference on CAD/CAM, Robotics and Factories of the Future, London, UK, pp. 773-778.

22. GHOSH, K., 1994, "Generation of feasible assembly sequences", Proceedings

of 10th International Conference on CAD/CAM, Robotics and Factories of the Future, Ottawa, Ontario, Canada pp. 99-104.

23. GOTTIPOLU, B.R. and GHOSH, K., 1994, "Recherche sur la planification du processus d'assemblage", Rapport technique, Project C.D.T. C172, École Polytechnique, Montréal, Québec, Canada.

24. HENRIOUD, J.M. and BOURJAULT, A., 1987 "Logic programming applied to assembly sequences determination", Proceedings of IASTED Conference on Robotics and Automation, Lugano, pp. 124-128.

25. HENRIOUD, J.M., BOURJAULT, A., and CHAPPE, D.,1987 "Élaboration des gammes d'assemblage : Approche composants", GAMI - Les Jours de la Productique.

26. HEEMSKERK, C.J.M. and VAN LUTTERVELT, C.A., 1989 "The use of heuristics in assembly sequence planning", Annals of CIRP, 38(1), 37-40.

27. HOFFMAN, R., 1989 "Automated assembly in a CSG domain", IEEE International Conference on Robotics and Automation, Scottsdale, Arizona, USA, pp. 210-215.

28. HOMEM DE MELLO, L.S. and SANDERSON, A.C., 1990 "AND/OR graph representation of assembly plans", IEEE Transactions on Robotics and Automation, 6(2), 188-199.

29. HOMEM DE MELLO, L.S. and SANDERSON, A.C., 1990 "Evaluation and selection of assembly plans", Proceedings of IEEE International Conference on Robotics and Automation, Las Alamitos, California, USA, pp. 1588-1593.

30. HOMEM DE MELLO, L.S. and SANDERSON, A.C., 1991 "Representation of mechanical assembly sequences", IEEE Transactions on Robotics and Automation, 7(2), 211-227.

31. HOMEM DE MELLO, L.S. and SANDERSON, A.C., 1991 "A correct and complete algorithm for the generation of mechanical assembly sequences", IEEE Transactions on Robotics and Automation, 7(2), 228-240.

32. HUANG, Y.F. and LEE, C.S.G., 1990 "An automatic assembly planning system", Proceedings of IEEE International Conference on Robotics and Automation, Las Alamitos, California, USA, pp. 1594-1599.

33. HUANG, Y.F. and LEE, C.S.G., 1991 "A framework for knowledge-based assembly planning", Proceedings of IEEE International Conference on Robotics and Automation, Sancramento, California, USA, pp. 599-604.

34. KHOSLA, P.K. and MATTIKALI, R., 1989 "Determining the assembly sequence from a 3-D model", Journal of Mechanical Working Technology, 20, 153-162.

35. KLEIN, C.J., 1987 "Generation and evaluation of assembly sequence alternatives", M.S. Thesis, Mechanical Engineering Department, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA.

36. KLEIN, C.M. and RHEE, H.K., 1991 "An assembly sequence generation procedure", Computers and Industrial Engineering, 21(1-4), 63-66.

37. KO, H. and LEE, K., 1982 "Automatic assembling procedure generation form mating conditions", Computer-Aided Design, 14(1), 3-10.

38. KRISHNAN, S.S. and SANDERSON, A.C., 1991 "Reasoning about geometric constraints for assembly sequence planning", Procedings of IEEE International Conf. on Robotics & Automation, Sancramento, California, USA, pp. 776-782.

39. LAPERRIERE, L. and ELMARAGHY, H.A., 1991 "Automatic generation of robotic assembly sequences", International Journal of Advanced Manufacturing Technology, 6(4), 299-316.

40. LEE, K. and GOSSARD, D.C., 1985 "A hierarchical data structure for representing assemblies: part 1", Computer-Aided Design, 17(1), 15-19.

41. LEE, S. and SHIN, Y.G., 1990 "Assembly planning based on subassembly extraction", Proceedings of IEEE International Conference on Robotics and Automation, Las Alamitos, California, USA, pp. 1606-1611.

42. LI, R.K. and HWANG, C.L., 1992 "A framework for automatic DFA system development", Computers and Industrial Engineering, 22(4), 403-413.

43. LIBERMAN, L.T. and WESLEY, M.A., 1977 "AUTOPASS: An automatic programming system for computer controlled assembly", IBM Journal of Research and Development, 21(4), 321-333.

44. LIN, A.C. and CHANG, T.C., 1993 "An integrated approach to automated assembly planning for three-dimensional mechanical products", International

Journal of Production Research, 31(5), 1201-1227.

45. LIN, A.C., and CHANG, T.C., 1993, "3D MAPS: Three-dimensional mechanical assembly planning system", Journal of Manufacturing Systems, 12(6), 437-456.

46. LIU, Y. and POPPLESTONE, R.J., 1989 "Planning for assembly from solid models", Proceedings of IEEE International Conference on Robotics and Automation, Scottsdale, Arizona, USA, pp. 222-227.

47. LUI, M.M., 1988 "Generation and evaluation of mechanical assembly sequences using the liaison-sequence method", M.S. Thesis, Dept. Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA, USA.

48. MAGAD, E. L., 1972 "Cooperative manufacturing research", Industrial Engineering, 4(1), 36-40.

49. MANTYALA, M., 1988 "An introduction to solid modeling", Computer Science Press, Rockville, MD, USA.

50. MASCLE, C. and FIGOUR, J., 1990 "Methodological approach of sequences determination using the disassembly method", Proceedings of Rensselaer 2nd International Conference on CIM, Troy, NY, USA, pp. 483-490.

51. MORRIS, G.H. and HAYNES, L.S., 1987 "Robotic assembly by constraints", Proceedings of IEEE International Conference on Robotics and Automation, Raleigh, North Carolina, USA, pp. 1507-1515.

52. NEVINS, J.L., WHITNEY, D.E. and DE FAZIO, T.L., 1990 "Concurrent design of products and processes", McGraw-Hill, New York, USA.

53. NNAJI, O., CHU, J. and AKREP, M., 1992 "A scheme for CAD-based robot assembly task planning for CSG modeled objects", Journal of Manufacturing Systems, 7(2), 131-145.

54. OYAMA, M. and ABE, K., 1992 "Application of stepwise clustering method for the determination of efficient assembly sequence", IEEE Transactions on Systems, Man and Cybernetics, 22(2), 267-274.

55. POPPLESTONE, R.J., AMBLER, A.P. and BELLOS, I., 1978 "RAPT: A language for describing assemblies", The Industrial Robot, 5(3), 131-137.

56. PRENTING, T. and BATTAGLIN, R., 1964, "The precedence diagram: A tool for analysis in assembly line balancing", Journal of Industrial Engineering, 15(4), 208-213.

57. PU, P., 1992 "An assembly sequence generation algorithm using case-based search techniques", Proceedings of IEEE International Conference on Robotics and Automation, Nice, France, pp. 2425-2430.

58. REDDY, G.B. and GHOSH, K., 1992 "Selection and evaluation of assembly sequences", Proceedings of 2nd International Conference on Automation Technology, Taipe, Taiwan, Vol. 4, pp. 69-72.

59. REQUICHA, A.A.G. and VOELCKER, H.B., 1983 "Solid modeling: Current status and research conditions", IEEE Computer Graphics and Applications, 3(7), 25-37.

60. RIVEST, L., FORTIN, C. and MOREL. C., 1994 "Tolerancing a solid model with a kinematic formulation", Computer-Aided Design, 26(6), 465-475.

61. ROCHELEAU, D.N. and LEE, K., 1987 "System for interactive assembly modeling", Computer Aided Design, 19(2), 65-72.

62. ROSEN, K.H., 1988 "Discrete mathematics and its applications", Random House, New York, USA.

63. ROSSIGNAC, J.R., 1986 "Constraints in constructive solid geometry", Technical Report No. RC12356, IBM Corporation, White Plains, NY, USA.

64. SANDERSON, A.C., HOMEM DE MELLO, L.S. and ZHANG, H., 1990 "Assembly sequence planning", AI Magazine, Spring '90, 62-81.

65. SANTOCHI, M. and DINI, G., 1992 "Computer-aided planning of assembly operations: The selection of assembly sequences", Robotics and Computer-Integrated Manufacturing, 9(6), 439-446.

66. SEKIGUCHI, H., KOJIMA, T., INOUE, K., HONDA, T. and TAKEYAMA, H., 1983 "Study on automatic determination of assembly sequences", Annals of the CIRP, 32(1), 25-28.

67. SHPITALNI, M., ELBER, G. and LENZ, E., 1988 "Automatic assembly of three dimensional structures via connectivity graph", Annals of the CIRP, 38(1), 25-28.

68. SRIKANTH, S. and TURNER, J.U., 1990 "Toward a unified representation of mechanical assemblies", Engineering with Computers, 6(2), 103-112.

69. STURGES, R.H. and KILANI, M.I., 1992 "Towards an integrated design for an assembly evaluation and reasoning system", Computer-Aided Design Journal, 24(2), 67-79.

70. SUBRAMANI, A.K. and DEWHURST, P., 1991 "Automatic generation of disassembly sequences", Annals of the CIRP, 40(1), 115-118.

71. SWIFT, K. and REDFORD, A.H., 1980 "Design for assembly", Engineering, 220(7), 779-802.

72. TAYLOR, R.H., 1976 "A synthesis of manipulator control programs from task-level specifications", Stanford Artificial Intelligence Lab Memo-AIM282, Stanford University, California, USA.

73. TURNER, J.U., 1989 "Relative positioning of parts in assemblies using mathematical programming", Technical Report N. 89046, Rensselaer Design Research Center, Troy, New York, USA.

74. WARRATS, J.J., BONSCAHNCHER, N. and BRONSVOORT, W.F., 1992 "A semi-automatic sequence planner" Proceedings of IEEE International Conference on Robotics and Automation, Nice, France, pp. 2431-2438.

75. WOLTER, J.D., 1989 "On the automatic generation of assembly plans", Proceedings of IEEE International Conference on Robotics and Automation, Scottsdale, Arizona, USA, pp. 62-68.

76. WOLTER, J.D., 1991 "A combinatorial analysis of enumerative data structures for assembly planning", Proceedings of IEEE International Conference on Robotics and Automation, Sancramento, California, USA, pp. 611-618.

77. ZUSSMAN, E., SHOHAM, M. and LENZ, E., 1992 "A kinematic approach to automatic assembly planning", Manufacturing Review, 5(4), 293-304.

78. PADL-2 Users Manual, Vol. 1, The Production Automation Project, University of Rochester, Rochester, NY, USA, 1980

79. Design for Assembly Toolkit, Release 5.2, Boothroyd Dewhurst, Inc., Wakefield, Rhode Island, USA, 1991.

APPENDIX

```
      PROGRAM MAIN_2
      IMPLICIT REAL*8(a-h,o-z)
      INTEGER I,J,X,NO,NPARTS
      REAL*4 OUTV
      REAL XMAX,YMAX,ZMAX,INCRMNT
      CHARACTER*132 OUTS
      CHARACTER*120 STATMENT(100)
      CHARACTER*1 PARTS(30)
      CHARACTER*8 OUTFILE,INFILE
      INTEGER TFUNC(6),CFUNC(6)
C*****************************************************************
C     read the PADL_2 definition of the assembly stored in a file
C*****************************************************************
      WRITE(6,*)'GIVE THE INPUT FILE NAME'
      READ(6,23)INFILE
      WRITE(6,*)'GIVE THE INCREMENT VALUE'
      READ(6,9)INCRMNT
      WRITE(6,*)'GIVE THE MAX INCREMENT IN X-DIRECTION'
      READ(6,9)XMAX
      WRITE(6,*)'GIVE THE MAX INCREMENT IN Y-DIRECTION'
      READ(6,9)YMAX
      WRITE(6,*)'GIVE THE MAX INCREMENT IN Z-DIRECTION'
      READ(6,9)ZMAX
C     WRITE(6,*)'GIVE THE NUMBER OF THE FIRST COMPONENT, STARTING AND'
C     WRITE(6,*)'ENDING NUMBERS OF THE SECOND COMPONENT OF THE PAIRS'
C     WRITE(^,*)'TO BE EVALAUTED '
C      READ(6,*)I,J1,J2
    9 FORMAT(F6.2)
   23 FORMAT(A8)
      OPEN(8,FILE=INFILE,STATUS='OLD')
      READ(8,10)NO,NPARTS
   10 FORMAT(I3,I2)
      write(6,*)'NO:',NO,'  NPARTS:',NPARTS
      DO 15 X=1,NO
      READ(8,20)STATMENT(X)
   20 FORMAT(A120)
   15 CONTINUE
      DO 12 X=1,NPARTS
      READ(8,21)PARTS(X)
   21 FORMAT(A1)
   12 CONTINUE
      CLOSE(8,STATUS='KEEP')
      CALL P2INIT
      CALL PP2('SET PROP_ERR=0;',OUTS,OUTV)
      DO 25 X=1,NO
      CALL PP2(STATMENT(X),OUTS,OUTV)
   25 CONTINUE
C*****************************************************************
C     display the four views of final assembly
C*****************************************************************
      CALL PADL_VIEWS
      CALL PP2('CENTER ASMBLY; DISP ASMBLY:.Q3,.WHITE;',OUTS,OUTV)
      CALL PP2('UGB; DISP ASMBLY:.Q1,.TOPVIEW,.WHITE;',OUTS,OUTV)
      CALL PP2('UGB; DISP ASMBLY:.Q2,.YVIEW,.WHITE;',OUTS,OUTV)
      CALL PP2('UGB; DISP ASMBLY:.Q4,.XVIEW,.WHITE;',OUTS,OUTV)
C***************************************************************
C     Extraction of C & T functions into output file
C***************************************************************
      WRITE(6,*)'GIVE THE OUTPUT FILE NAME'
      READ(6,23)OUTFILE
      OPEN(10,FILE=OUTFILE,STATUS='UNKNOWN')
      WRITE(10,*)
      WRITE(10,*)'PAIRS     C-FUNCTIONS        T-FUNCTIONS'
      WRITE(10,*)'-----   ----------------   ----------------'
      WRITE(10,*)
      WRITE(6,*)
      WRITE(6,*)'PAIRS     C-FUNCTIONS        T-FUNCTIONS'
      WRITE(6,*)'-----   ----------------   ----------------'
      WRITE(6,*)
      DO 35 I=1,NPARTS
      DO 35 J=1,NPARTS
      IF (I .GE. J) GOTO 35
      CALL CONTACT(PARTS,NPARTS,CFUNC,1,'MOVX',INCRMNT,I,J)
      CALL CONTACT(PARTS,NPARTS,CFUNC,2,'MOVY',INCRMNT,I,J)
.      CALL CONTACT(PARTS,NPARTS,CFUNC,3,'MOVZ',-INCRMNT,I,J)
      CALL CONTACT(PARTS,NPARTS,CFUNC,4,'MOVX',-INCRMNT,I,J)
      CALL CONTACT(PARTS,NPARTS,CFUNC,5,'MOVY',-INCRMNT,I,J)
      CALL CONTACT(PARTS,NPARTS,CFUNC,6,'MOVZ',INCRMNT,I,J)
      CALL TRANSLATION(PARTS,NPARTS,TFUNC,1,'MOVX',INCRMNT,XMAX,I,J)
      CALL TRANSLATION(PARTS,NPARTS,TFUNC,2,'MOVY',INCRMNT,YMAX,I,J)
      CALL TRANSLATION(PARTS,NPARTS,TFUNC,3,'MOVZ',-INCRMNT,ZMAX,I,J)
      CALL TRANSLATION(PARTS,NPARTS,TFUNC,4,'MOVX',-INCRMNT,XMAX,I,J)
      CALL TRANSLATION(PARTS,NPARTS,TFUNC,5,'MOVY',-INCRMNT,YMAX,I,J)
      CALL TRANSLATION(PARTS,NPARTS,TFUNC,6,'MOVZ',INCRMNT,ZMAX,I,J)
      WRITE(10,*)'(',PARTS(I),',',PARTS(J),')  ',CFUNC(1),' ',CFUNC(2),
     +     ' ',CFUNC(3),' ',CFUNC(4),' ',CFUNC(5),' ',CFUNC(6),'  ',
     +     TFUNC(1),' ',TFUNC(2),' ',TFUNC(3),' ',TFUNC(4),' ',
     +     TFUNC(5),' ',TFUNC(6)
      WRITE(6,*)'(',PARTS(I),',',PARTS(J),')  ',CFUNC(1),' ',CFUNC(2),
     +     ' ',CFUNC(3),' ',CFUNC(4),' ',CFUNC(5),' ',CFUNC(6),'  ',
     +     TFUNC(1),' ',TFUNC(2),' ',TFUNC(3),' ',TFUNC(4),' ',
     +     TFUNC(5),' ',TFUNC(6)
   35 CONTINUE
      CLOSE(10,STATUS='KEEP')
      END
C
C*******************************************************************
C     subroutine for T functions
C*******************************************************************
      SUBROUTINE TRANSLATION(PARTS,NPARTS,FUNC,DIR,MOBL,INCRMNT,MAX,I,J)
      INTEGER I,J,K,P,NPARTS
      REAL*4 OUTV
      REAL VOL,MV,MAX,INCRMNT
      CHARACTER*132 OUTS
      CHARACTER*6 TITLE
      CHARACTER*120 INTERS
      CHARACTER*1 PARTS(30)
      INTEGER FUNC(6),DIR
      CHARACTER*4 MOBL
      FUNC(DIR)=1
      MV = 0.0
   11 MV=MV+INCRMNT
      ENCODE(120,140,INTERS)PARTS(I),PARTS(J),MOBL,MV
  140 FORMAT('INTERS = ',A1,' INT ',A1,' MBY ',A4,'=',F7.2,';')
      DECODE(120,150,INTERS)INTERS
  150 FORMAT(A120)
      CALL PP2(INTERS,OUTS,OUTV)
      OPEN(7,FILE='INTER_PROP',STATUS='UNKNOWN')
      CALL PP2('SET PROP_UNIT=7;',OUTS,OUTV)
      CALL PP2('SET PROP_LEVEL=4;',OUTS,OUTV)
      CALL PP2('PROP INTERS;',OUTS,OUTV)
      write(6,*)'T  MOBL:',MOBL,'  BY ',MV
      CLOSE(7,STATUS='KEEP')
      OPEN(7,FILE='INTER_PROP',STATUS='OLD')
      DO 31 K=1,14
      READ(7,30)P
   30 FORMAT(I1)
   31 CONTINUE
      READ(7,40)TITLE,VOL
   40 FORMAT(1X,A6,2X,E11.4)
      IF (TITLE .EQ. 'VOLUME') GOTO 22
      READ(7,50)VOL
   50 FORMAT(9X,E11.4)
   22 CLOSE(7,STATUS='KEEP')
      IF (VOL .EQ. 0.0) THEN
       IF (ABS(MV) .LT. MAX) GOTO 11
      ELSE
       FUNC(DIR)=0
      ENDIF
      RETURN
      END
C*******************************************************************
C     subroutine for C functions
C*******************************************************************
      SUBROUTINE CONTACT(PARTS,NPARTS,FUNC,DIR,MOBL,INCRMNT,I,J)
      INTEGER I,J,K,P,NPARTS
      REAL*4 OUTV
      REAL VOL,INCRMNT
      CHARACTER*132 OUTS
      CHARACTER*6 TITLE
      CHARACTER*120 INTERS
      CHARACTER*1 PARTS(30)
      INTEGER FUNC(6),DIR
      CHARACTER*4 MOBL
      FUNC(DIR)=0
      ENCODE(120,140,INTERS)PARTS(I),PARTS(J),MOBL,INCRMNT
  140 FORMAT('INTERS = ',A1,' INT ',A1,' MBY ',A4,'=',F6.2,';')
      DECODE(120,150,INTERS)INTERS
  150 FORMAT(A120)
      CALL PP2(INTERS,OUTS,OUTV)
      OPEN(7,FILE='INTER_PROP',STATUS='UNKNOWN')
      CALL PP2('SET PROP_UNIT=7;',OUTS,OUTV)
      CALL PP2('SET PROP_LEVEL=4;',OUTS,OUTV)
      CALL PP2('PROP INTERS;',OUTS,OUTV)
      CLOSE(7,STATUS='KEEP')
      OPEN(7,FILE='INTER_PROP',STATUS='OLD')
      DO 31 K=1,14
      READ(7,30)P
   30 FORMAT(I1)
   31 CONTINUE
      READ(7,40)TITLE,VOL
   40 FORMAT(1X,A6,2X,E11.4)
      IF (TITLE .EQ. 'VOLUME') GOTO 22
      READ(7,50)VOL
   50 FORMAT(9X,E11.4)
   22 CLOSE(7,STATUS='KEEP')
      IF (VOL .EQ. 0.0) THEN
      FUNC(DIR)=0
      ELSE
       FUNC(DIR)=1
      ENDIF
      RETURN
      END
C*******************************************************************
C     subroutine to initialse PADL views
C*******************************************************************
      SUBROUTINE PADL_VIEWS()
      CHARACTER*132 OUTS
      REAL*4 OUTV
      OUTS='DUMMY'
C
C     ***** Set line colors******
C
      CALL PP2('.RED=(LINE_COLOR=2);',OUTS,OUTV)
      CALL PP2('.GREEN=(LINE_COLOR=4);',OUTS,OUTV)
      CALL PP2('.YELLOW=(LINE_COLOR=3);',OUTS,OUTV)
      CALL PP2('.WHITE=(LINE_COLOR=1);',OUTS,OUTV)
C
C     ***** Define the views to diplay the components*****
C
      CALL PP2('.TOPVIEW=(VIEW_TYPE=0,DIRECTION=LAB);',OUTS,OUTV)
      CALL PP2('.XVIEW=(VIEW_TYPE=0,DIRECTION=DEGY=90);',OUTS,OUTV)
      CALL PP2('.YVIEW=(VIEW_TYPE=0,DIRECTION=DEGX=90);',OUTS,OUTV)
C
C     *****Define the windows to diplay the different views*****
C
      CALL PP2('.Q1=DEVICE=000012;',OUTS,OUTV)
      CALL PP2('.Q2=DEVICE=000013;',OUTS,OUTV)
      CALL PP2('.Q3=DEVICE=000015;',OUTS,OUTV)
      CALL PP2('.Q4=DEVICE=000014;',OUTS,OUTV)
C
C     *****Clear all the windows******
C
      WRITE(6,*)'CLEARING THE WINDOWS'
      CALL PP2('SET .Q1;ERASE;',OUTS,OUTV)
      CALL PP2('SET .Q2;ERASE;',OUTS,OUTV)
      CALL PP2('SET .Q3;ERASE;',OUTS,OUTV)
      CALL PP2('SET .Q4;ERASE;',OUTS,OUTV)
C
      RETURN
      END
```

```
28 6
CN1 = CON(H=12,D=6) MBY DEGY=90
CL1 = CYL(H=4, D=5) MBY DEGY=90
CL2 = CYL(H=10, D=1.5) MBY DEGY=90
CL3 = (CL1 UN CL2) MBY MOVX=12
L1 = CYL(H=1.5, D=0.5) MBY DEGY=90, MOVX=20.5
H = CN1 UN CL3 DIF L1
CL4 = CYL(H=90, D=3)
CL5 = CYL(H=90, D=1.5)
T = (CL4 DIF CL5) MBY DEGY=90, MOVX=16
L2 = CYL(H=1.45, D=0.45) MBY DEGY=90, MOVX=20.55
L3 = CYL(H=80, D=1.5) MBY DEGY=90, MOVX=22
I = L2 UN L3
CN3 = (CON(H=16, D=8) MBY DEGY=90) DIF CN1
CL6 = CYL(H=100, D=8) MBY DEGY=90, MOVX=16
NL1 = CN3 UN CL6
CL7 = CYL(H=104, D=5) MBY DEGY=90, MOVX=12
B = NL1 DIF CL7
CL8 = CYL(H=7, D=5) MBY DEGY=90
CL9 = CYL(H=1, D=8) MBY DEGY=90, MOVX=6
CL10 = CL8 UN CL9
U = CL10 MBY MOVX=110
CN4 = CON(H=50, D=8) DIF CON(H=25, D=4)
CN5 = CON(H=64, D=10) DIF CON(H=38, D=6)
CN6 = CN5 DIF (CN4 MBY MOVZ=14)
CL11 = CYL(H=8, D=10) DIF CYL(H=8, D=8)
CL12 = CL11 MBY MOVZ=64
C = (CN6 UN CL12) MBY DEGY=90, MOVX=-48.5
ASMBLY = H ASB T ASB I ASB B ASB U ASB C
C
B
U
H
T
I
```

```
49 11
B1 = BLO(X=22.5, Y=1.38, Z=18.5)
B2 = BLO(X=22.26, Y=1.5, Z=18.26) MBY MOVX=0.12, MOVY=0.12, MOVZ=0.12
B3 = BLO(X=0.32, Y=0.13, Z=4.0) MBY MOVX=1.0, MOVZ=1.0
BL1 = BLO(X=0.32, Y=0.13, Z=4.0) MBY MOVX=21.18, MOVZ=1.0
C1 = CYL(H=0.2, D=0.2) MBY DEGX=-90, MOVX=1.0, MOVZ=17.0
CL1 = CYL(H=0.2, D=0.2) MBY DEGX=-90, MOVX=21.5, MOVZ=17.0
B4 = BLO(X=12.5,Y=0.13, Z=5.5) MBY MOVX=5.0, MOVZ=6.5
A = B1 DIF B2 DIF B3 DIF B4 DIF C1 DIF BL1 DIF CL1
B = BLO(X=13.5, Y=0.14, Z=6.5) MBY MOVX=4.5, MOVY=0.12, MOVZ=6.0
B5 = BLO(X=13.24, Y=0.9, Z=6.24) MBY MOVX=4.63, MOVZ=6.13, MOVY=0.261
B6 = BLO(X=13.02, Y=1.05, Z=6.02) MBY MOVX=4.74, MOVZ=6.24, MOVY=0.261
D = B5 DIF B6
B7 = BLO(X=17.0, Y=0.9, Z=14.5) MBY MOVX=2.7, MOVY=0.261, MOVZ=2.0
E = B7 DIF B5
B9 = BLO(X=17.72, Y=1.15, Z=14.98) MBY MOVX=2.39, MOVY=0.12, MOVZ=1.76
B10 = BLO(X=17.5, Y=1.04, Z=14.76) MBY MOVX=2.5, MOVY=0.12, MOVZ=1.87
F = B9 DIF B10 DIF B6
B11 = BLO(X=22.74, Y=1.5, Z=18.74) MBY MOVX=-0.12, MOVZ=-0.12
B12 = BLO(X=22.5, Y=1.38, Z=18.5)
B13 = BLO(X=20.0, Y=1.5, Z=16.0) MBY MOVX=1.25, MOVZ=1.25
B14 = BLO(X=22.74, Y=1.64, Z=0.12) MBY MOVX=-0.12, MOVZ=-0.12
B15 = BLO(X=22.74, Y=0.12, Z=0.2) MBY MOVX=-0.12, MOVY=1.64
C2 = CYL(H=0.2, D=0.4) MBY DEGX=-90, MOVX=1.0, MOVZ=17.0, MOVY=1.38
CL2 = CYL(H=0.2, D=0.4) MBY DEGX=-90, MOVX=21.5, MOVZ=17.0, MOVY=1.38
H = B11 DIF B12 DIF B13 UN B14 UN B15 DIF C2 DIF CL2
C3 = CYL(H=1.37, D=0.4) UN CYL(H=1.26, D=0.5)
C4 = C3 DIF CYL(H=1.37, D=0.2)
CL3 = C4 MBY DEGX=-90, MOVX=1.0, MOVZ=17.0, MOVY=0.12
CL4 = C4 MBY DEGX=-90, MOVX=21.5, MOVZ=17.0, MOVY=0.12
G = CL3 UN CL4
BL2 = BLO(X=22.74, Y=0.14, Z=18.62) MBY MOVX=-0.12, MOVY=1.5
CL5 = CYL(H=0.2, D=0.2) MBY DEGX=-90, MOVX=1.0, MOVZ=17.0, MOVY=1.5
CL6 = CYL(H=0.2, D=0.2) MBY DEGX=-90, MOVX=21.5, MOVZ=17.0, MOVY=1.5
I = BL2 DIF CL5 DIF CL6
C5 = CYL(H=0.3,D=0.21) MBY DEGX=-90, MOVX=0.5, MOVZ=0.5
B16 = BLO(X=1.0, Y=2.0, Z=1.0) DIF C5
B17 = BLO(X=21.5, Y=1.0, Z=1.0) MBY MOVY=1.0
B18 = B16 MBY MOVX=20.5
B19 = B16 UN B17 UN B18
J = B19 MBY MOVX=0.5, MOVY=1.65, MOVZ=16.5
C6 = CYL(H=2.1, D=0.2) UN CYL(H=0.16, D=0.4)
K = C6 MBY DEGX=-90, MOVX=1.0,MOVZ=17.0, MOVY=-0.16
B20 = BLO(X=1.0, Y=0.75, Z=10.0) MBY MOVX=0.66, MOVZ=0.12, MOVY=0.12
B21 = BLO(X=0.32, Y=3.0, Z=3.0) MBY MOVX=1.0, MOVZ=1.0, MOVY=-1.52
B22 = BLO(X=0.6, Y=0.5, Z=6.0) MBY MOVX=0.86, MOVZ=10.12, MOVY=0.12
BL3 = B20 UN B21 UN B22
BL4 = BL3 MBY MOVX=20.18
C = BL3 UN BL4
ASMBLY = A ASB B ASB C ASB D ASB E ASB F ASB G ASB H ASB I ASB J ASB K
A
B
C
D
E
F
G
H
I
J
K
```