UNIVERSITÉ DE MONTRÉAL

INTELLIGENT COMPUTER-AIDED PROCESS PLANNING FOR ROTATIONALLY SYMMETRICAL PARTS USING NEURAL NETWORK AND EXPERT SYSTEM

SANKHA DEB
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIAE DOCTOR (Ph.D.)
(GÉNIE ÉLECTRIQUE)
JUIN 2005

Your file Votre référence
ISBN: 978-0-494-16990-2
Our file Notre référence
ISBN: 978-0-494-16990-2

To my parents and my brother

ACKNOWLEDGMENTS

First and foremost, the author expresses sincere thanks and a deep sense of gratitude to the author's esteemed teacher and thesis adviser, Professor Kalyan Ghosh for his invaluable advice, inspiring guidance, motivation and professional and personal advice, and kind assistance at all stages of the present work. He also made available necessary funds for undertaking this research for which I am grateful to him. Professor Ghosh introduced me to the research topic of Computer-Aided Process Planning and Computer Integrated Manufacturing in general. He gave me constant encouragement by always demonstrating interest in my research during my stay in Montreal as a graduate student.

The author also would like to thank Professor Marek Balazinski of the Mechanical Engineering Department of École Polytechnique and Dr. Chandra Devireddy of Key Solutions Inc., Santa Clara, California USA for their constructive comments and suggestions during the presentation of the research proposal. The author also thankfully acknowledges Professor Clement Fortin of the Mechanical Engineering Department, Professor Jean-Jules Brault of the Electrical Engineering Department of École Polytechnique, Dr. Warren Hall of Pratt and Whitney Canada and Professor Jami Shah of the Department of Mechanical and Aerospace Engineering of Arizona State University, USA for their course materials, which helped the author to proceed with the research work. The author would also like to thank Mr. Paul Brunelle and Mr. Benoit Forest, Computer Technician and System Administrator of the Department of Mathematics and Industrial Engineering of École Polytechnique for their support in administrating the laboratory computer system. The author would also like to thank Mr. Brian Kett, Ms. Nina Fitzsimons, Ms. Vicky Berry and Mr. Trevor Cole of Neusciences, UK for their prompt technical support with the NeuFrame software package.

The author is also thankful to Dr. Soumitra Paul of the Mechanical Engineering Department of Indian Institute of Technology, Kharagpur, India for his constructive comments. The author gratefully thanks Professor A. B. Chattopadhyay, who was his thesis adviser for the Master's degree and Professor P. K. Mishra of the Mechanical Engineering Department of Indian Institute of Technology, Kharagpur for their constant source of encouragement and motivation to pursue the research work in the said field.

The author also thanks his colleagues, in particular Mr. Eugen Coman, Ph.D. candidate, and Mr. Pierre-Emanuel Coulon and Mr. Raul Parra, Master's degree students of the Department of Mathematics and Industrial Engineering of École Polytechnique for being very helpful during present investigation.

To all other professors and the research colleagues in the Department of Mathematics and Industrial Engineering at École Polytechnique the author would also like to express thanks. Sincere thanks are also due to the staff members of the department for their support and encouragement during the research work.

The author is ever indebted to his parents, Mrs. Sima Deb and Professor Satyaranjan Deb, and his brother, Dr. Soumya Deb for their blessings and support. It is to them the author dedicates this work.

RÉSUMÉ

Le travail de recherche présenté dans cette thèse vise à explorer les applications des différentes techniques d'intelligence artificielle afin d'automatiser deux des tâches importantes de la prise de décision dans les systèmes de génération automatique des gammes d'usinage de type génératif, notamment la sélection des opérations d'usinage et la planification du montage pour les pièces cylindriques.

Une revue de la littérature est présentée sur les différentes approches génératives de la génération des gammes d'usinage, développées par les autres chercheurs, pour automatiser la tâche de la sélection des opérations d'usinage en utilisant l'arbre de décision, le système expert et le réseau de neurones artificiels. Les avantages et les inconvénients de chacune des approches ont été identifiés. La revue de la littérature montre que l'approche basée sur le réseau de neurones peut surmonter plusieurs limitations des approches basées sur l'arbre de décision et le système expert. Malgré cela, une limitation des approches basées sur le réseau de neurones développées par les autres chercheurs est qu'il n'y a aucune directive pour choisir les formes d'entrée pour les exemples utilisés pour l'entraînement du réseau de neurones. De plus une question qui n'a pas été suffisamment explorée par les autres chercheurs est qu'on pourrait profiter de la connaissance antérieure du domaine de la sélection des opérations d'usinage. En outre, la plupart des modèles de réseau de neurones tendent à recommander une seule séquence des opérations d'usinage pour usiner chacun des types de particularités de la pièce. En considérant tout ce qui est mentionné ci-dessus, l'auteur a développé une méthodologie pour automatiser la tâche de la sélection des opérations d'usinage en utilisant une approche basée sur le réseau de neurones artificiels. Un progiciel intitulé NeuFrame est utilisé sur un micro-ordinateur pour simuler l'opération de réseau de neurones. Le réseau de neurones a été d'abord pré-structuré avec la connaissance antérieure de domaine sous forme de règles heuristiques. Deux formes de représentation des données d'entrée au réseau de neurones ont été développées. La représentation externe est utilisée pour entrer les valeurs spécifiques des variables de la prise de décision (notamment le type des particularités de la pièce et ses attributs, comme par exemple le diamètre ou la largeur, la tolérance et la finition de la surface) au réseau neuronal. La représentation interne convertit les valeurs spécifiques des variables en des ensembles qui correspondent à diverses catégories de particularités géométriques et leurs attributs telles que représentées dans la partie « Si » des règles heuristiques mentionnées précédemment. La couche d'entrée de neurones dans le réseau neuronal a été conçue de telle manière qu'un nœud neuronal soit assigné à chaque type de particularités et à l'ensemble de ses attributs. Dans la couche de sortie de réseau neuronal, un noeud est assigné à chacune des séquences réalisables parmi des opérations d'usinage qu'on retrouve dans la partie conséquente <<Alors>> des règles heuristiques. L'algorithme de rétropropagation a été employé comme mécanisme d'apprentissage pour entraîner le réseau neuronal. Une méthode systématique est illustrée pour choisir des formes d'entrée pour les exemples d'entraînement du réseau neuronal. Les règles heuristiques mentionnées précédemment ont été employées pour agir comme directives pendant la préparation des exemples d'entraînement de réseau neuronal; les formes d'entrée sont choisies de telle manière qu'elles activent une ou plusieurs de ces règles heuristiques. La modification de la base de connaissance peut être accomplie facilement et rapidement tout simplement par le réentraînement du réseau neuronal avec un nouvel ensemble d'exemples d'entraînement. Deux exemples des pièces cylindriques ont été analysés en utilisant l'approche de réseau de neurones proposée par l'auteur pour démontrer son potentiel d'application dans le vrai environnement de fabrication. Les possibilités de l'extension de la méthodologie proposée et les directions prometteuses pour le travail de recherche dans le futur ont été indiquées. Par cette méthodologie basée sur le réseau de neurones, la tâche de la sélection des opérations d'usinage peut être effectué en investissant peu de temps, la rendant de ce fait très efficace et rentable pour des applications industrielles.

Une revue de la littérature est présentée sur les différentes approches génératives de la génération des gammes d'usinage, développées par les autres chercheurs, pour automatiser la tâche de la planification du montage en utilisant les algorithmes et les graphiques, le système expert, la logique floue et le réseau de neurones artificiels. Les avantages et les inconvénients de chacune des approches ont été identifiés. La revue de la littérature montre que l'approche basée sur le système expert possède beaucoup de potentiel pour automatiser cette activité. Mais la plupart des efforts de recherche pour automatiser la tâche de la planification du montage en utilisant l'approche de système expert ont été limités au domaine des pièces prismatiques et leur potentiel pour les pièces cylindriques n'a pas été suffisamment exploré. De plus, dans la plupart des approches de système expert qui ont été rapportées par les autres chercheurs, un mélange de la méthode de système expert et d'une méthode algorithmique a été adopté, afin d'automatiser les différentes tâches de la planification du montage. Une limitation importante est que le système qui en résulte tend à être inflexible et répond mal aux nouvelles situations. En considérant tout ce qui est mentionné ci-dessus, l'auteur a développé une méthodologie pour automatiser les différentes tâches de la planification du montage en utilisant une approche basée sur le système expert. Elle a été implantée sur un micro-ordinateur en utilisant le progiciel CLIPS. Afin de représenter les données d'entrée pour le système expert, un format de représentation a été conçu en utilisant le <<Template>>, qui est le format généralement utilisé pour la représentation de données d'entrée dans CLIPS. Une base de connaissance comprenant les règles a été développée pour chacun des problèmes, notamment le groupement des opérations, l'ordonnancement des opérations et la sélection des surfaces de la pièce sur lesquelles il faut la localiser et la fixer pour chaque montage. Pour effectuer le groupement des opérations, un ensemble de règles a été développé, qui est capable de grouper les opérations d'usinage dans les différents montages. Les facteurs qu'on a considérés sont la direction d'accès d'outil pour usiner chacune des particularités et les relations de tolérances entres les particularités. Pour l'ordonnancement des opérations, un ensemble de règles a été développé, qui est capable de déterminer les séquences des opérations dans chaque

montage selon les contraintes de priorité entres les différentes particularités ainsi que la logique de fabrication pour l'ordonnancement des opérations. Un ensemble de règles, basé sur des principes heuristiques a été employé pour la sélection des surfaces de la pièce sur lesquelles il faut la localiser et la fixer pour chaque montage. La modification de la base de connaissance peut être faite en modifiant tout simplement les règles dans la base de connaissance du système expert. Les nouvelles connaissances peuvent être facilement acquises par les systèmes experts en introduisant de nouvelles règles à sa base de connaissance. Deux exemples des pièces cylindriques ont été analysés en utilisant l'approche de système expert proposée par l'auteur pour démontrer son potentiel d'application. Les possibilités de l'extension de la méthodologie proposée et les directions prometteuses pour le travail de recherche dans le futur ont été indiquées. Par cette méthodologie basée sur le système expert, la tâche de la planification du montage peut être effectuée en investissant peu de temps, la rendant très efficace et rentable pour des applications industrielles.

ABSTRACT

The research work reported in this thesis is aimed at exploring possible applications of different Artificial Intelligence (AI) techniques for automating two of the important decision making tasks in generative Computer-Aided Process Planning (CAPP) systems, namely the selection of machining operations and the set-up planning in the case of rotationally symmetrical parts.

The relevant literature on previous research work for automating the machining operations selection by decision trees, expert systems and neural networks has been reviewed, highlighting their contributions and shortcomings. The literature review indicates that the neural networks based approaches can overcome several limitations of the decision trees and the expert system based approaches. However, inspite of the above advantages, a limitation observed with the neural network based approaches developed by previous researchers is that there are no guidelines for choosing the input patterns of training examples. Also an issue that has not been adequately addressed by previous researchers is that whether any prior domain knowledge on machining operations selection could be taken advantage of. Further, most of the previously developed neural network models tend to recommend a single machining operation sequence for a given feature of the part. Keeping the above in mind, a neural network based methodology has been developed for selection of all the possible alternative operation sequences for machining a given feature of the part. The PC based software package NeuFrame Version 4 (2000) has been used to simulate the neural network operation. The neural network has been pre-structured with prior knowledge on machining operations selection in the form of heuristic or thumb rules. It has been achieved by developing two forms of representation for the input data to the neural network. The external representation is used to enter the crisp values of the input decision variables (namely the feature type and its attributes such as diameter or width, tolerance and surface finish) to the neural network. The purpose of internal representation is to categorize the above crisp values into sets; these sets, in turn, correspond to all the possible different ranges of dimension, tolerance and surface finish, encountered in the antecedent 'IF' part of the thumb rules mentioned above. The input layer of the neural network has been designed in such a way that one neuronal node is allocated for each of the feature type and the above sets of feature attributes. The output layer of the neural network has been designed in such a way that one neuronal node is allocated to each of the various feasible machining operation sequences found in the consequent 'THEN' part of the thumb rules. The backpropagation algorithm has been used as the learning mechanism for the neural network. During the preparation of training examples, the thumb rules developed for selection of machining operation sequences have been used to serve as guidelines with the input patterns of training examples chosen in such a way that they activate one or more of these thumb rules. Any modification of the process planning knowledge base can be accomplished easily and quickly by merely retraining the network with a new set of training examples. The examples of some industrially-relevant rotationally symmetrical workpieces have been analyzed using the proposed approach to demonstrate its potential for use in the real manufacturing environment. The scope for further work and future research directions have been indicated. By this novel neural network based methodology, machining operations selection of workpieces of complex shapes can be handled by investing a very limited amount of time, making it attractive and cost effective for industrial applications.

A review of previous research work for automating the set-up planning by various approaches such as algorithms and graphs, expert systems, fuzzy logic and neural networks has been conducted, highlighting their contributions and shortcomings. The literature review indicates that the expert system has a lot of potential for solving the problem of set-up planning. However, most of the research efforts so far have been limited to the prismatic parts domain. Their potential for application in set-up planning of rotational parts has yet to be fully explored. Also, in most of the expert system based approaches that have been reported thus far by previous researchers, a mixture of the expert systems and some algorithmic approach has been adopted in order to solve the set-up planning problems. One shortcoming of it is that the resulting system tends to be inflexible, responding poorly to new situations. Keeping the above in mind, a rule based expert system approach has been developed by the author for solving the different set-up planning problems. It has been implemented on a PC by using the CLIPS rule-based expert system shell. In order to represent the input data for the expert system namely, the features present in the part and the operations for machining each feature, a format for representation has been developed by the author using template that is the commonly used input data representation format in CLIPS. To carry out set-up formation, a set of rules has been developed, which are capable of clustering the machining operations into set-ups taking into consideration the Tool Access Direction (TAD) of the corresponding features and the relative tolerance relationships between them. Further, a set of rules has been proposed to establish the various feature precedence constraints and to determine the operation sequence in each set-up, subject to the above precedence constraints as well as manufacturing logic for ordering the operations. Also a set of rules, based on heuristic principles developed by the author has been used for selection of the locating and clamping features in each set-up. Any modification of the current set-up planning knowledge can be done by simply modifying the rules in the knowledge base of the expert system. Also new knowledge can be easily acquired by the expert systems through introduction of new rules to its knowledge base. The examples of some rotationally symmetrical workpieces have been analysed using the proposed approach to demonstrate its potential for application. The scope for further work and future research directions have been given. By the expert system based methodology developed in this work, the set-up planning can be accomplished automatically by investing a very limited amount of time, making it attractive and cost effective for industrial applications.

CONDENSÉ EN FRANCAIS

C. 1 Introduction

La génération des gammes d'usinage est une activité importante de fabrication qui interprète les spécifications de la conception d'une pièce, et détermine systématiquement les étapes détaillées pour fabriquer la pièce, selon les spécifications, en utilisant les différents ressources de fabrication (par exemple, les outillages, les machines-outils et les autres équipements) qui sont disponibles dans l'usine et leurs capacités technologiques. Elle aide à intégrer le système de la Conception Assistée par Ordinateur (CAO) et le système de la Fabrication Assistée par Ordinateur (FAO) par l'établissement d'un lien entre les deux en traduisant les spécifications de la conception aux étapes détaillées de la fabrication. Par conséquent, elle détermine aussi le coût de la fabrication, la qualité de la pièce fabriquée, le délai pour l'exécution de la fabrication et l'efficacité de la production. Traditionnellement pendant la génération manuelle des gammes d'usinage, le planificateur prend les différentes décisions basées sur son intuition et les règles heuristiques d'après ses expériences et ses qualifications telles que sa capacité d'interpréter les spécifications de la pièce, d'analyser les besoins pour sa fabrication, sa connaissance des différents procédés d'usinage et les ressources de fabrication (par exemple, les outillages, les machines-outils, etc.) disponibles dans l'usine, ainsi que ses capacités technologiques et sa capacité de comprendre les interactions entre la pièce, la méthode de fabrication, la qualité et le coût. Pour surmonter les limitations possibles lors de la génération manuelle des gammes d'usinage, les différentes méthodes de la <<génération automatique des gammes d'usinage>> (CAPP) ont été développées par les chercheurs. L'idée principale est d'utiliser la puissance des ordinateurs pour aider à automatiser des différentes tâches de la prise des décisions lors de la génération des gammes d'usinage. Le travail de recherche présenté dans cette thèse vise à explorer les applications des différentes techniques d'intelligence artificielle afin d'automatiser deux des tâches importantes de la prise de décision dans les systèmes de génération automatique des gammes d'usinage de type génératif, notamment la sélection des opérations d'usinage et la planification du montage pour les pièces cylindriques.

C. 2 Organisation de la thèse

Cette thèse est organisée comme suit. Dans le chapitre 1, la motivation de ce travail de recherche, les objectifs, la méthodologie de recherche et une introduction générale sur les différentes approches de la génération automatique des gammes d'usinage sont brièvement présentés. Dans le chapitre 2, une revue de la littérature est présentée sur les différentes approches génératives de la génération des gammes d'usinage, développées par les autres chercheurs, pour automatiser les deux tâches suivantes : la sélection des opérations d'usinage et la planification du montage. La revue de la littérature est classifiée selon le type d'approche générative, ceux basés sur l'arbre de décision, les algorithmes et les graphiques, les techniques d'intelligences artificielles comme par exemple le système expert, la logique floue et le réseau de neurones artificiels. Les avantages et les inconvénients de chacune des approches ont été identifiés.

Dans le chapitre 3, la méthodologie développée par l'auteur est décrite pour automatiser la tâche de la sélection des opérations d'usinage en utilisant une approche basée sur le réseau de neurones artificiels. Les détails de la méthodologie qui sont élaborées incluent le rassemblement de la connaissance nécessaire pour la sélection des opérations d'usinage afin de formuler les règles heuristiques, la topologie du réseau de neurones, le format de la représentation des variables d'entrée et de sortie pour la prise de décision, et la méthode de l'entraînement et de la validation du réseau de neurones. Un exemple d'une pièce cylindrique est analysé en utilisant l'approche proposée par l'auteur pour démontrer son potentiel d'application, et les résultats sont aussi présentés. De plus les différences entre la méthodologie de réseau de neurones développé par l'auteur et ceux qui ont été adoptés par les autres chercheurs ont été indiquées avec les avantages importants de l'approche proposée.

Dans le chapitre 4, la méthodologie développée par l'auteur est présentée pour automatiser la tâche de la planification du montage. Les détails de la méthodologie qui sont élaborés incluent le développement de la structure du système expert, la base de données, la base de connaissance et le moteur d'inférence pour les différents problèmes de la planification du montage. Un exemple d'une pièce cylindrique est analysé en utilisant l'approche proposée par l'auteur pour démontrer son potentiel d'application, et les résultats sont aussi présentés. Les différences entre l'approche développée par l'auteur et celles qui ont été proposées par les autres chercheurs ont été discutées et les avantages de l'approche développée par l'auteur sont précisés.

Le chapitre 5 conclut la thèse en résumant le travail de recherche qui a été fait par l'auteur et les conclusions, en identifiant les contributions principales et en explorant les possibilités de l'extension de la méthodologie proposée et les directions prometteuses pour le travail de recherche dans le futur.

C. 3 Automatisation de la sélection des opérations d'usinage par le réseau de neurones artificiels

Pour le problème de l'automatisation de la sélection des opérations d'usinage, la revue de la littérature montre que l'approche basée sur le réseau de neurones peut surmonter plusieurs limitations des approches basées sur l'arbre de décision et le système expert. Malgré cela, une limitation des approches basées sur le réseau de neurones développées par les autres chercheurs est qu'il n'y a aucune directive pour choisir les formes d'entrée pour les exemples utilisés pour l'entraînement du réseau de neurones. De plus une question qui n'a pas été suffisamment explorée par les autres chercheurs est qu'on pourrait profiter de la connaissance antérieure du domaine de la sélection des opérations d'usinage. En outre, la plupart des modèles de réseau de neurones tendent à recommander une seule séquence des opérations d'usinage pour usiner chacun des types de particularités de la pièce. En considérant tout ce qui est mentionné ci-dessus, les objectifs de ce travail de recherche sont définis comme suit:

a) développer une méthodologie de la <<génération automatique des gammes d'usinage>> pour automatiser la tâche de la sélection de toutes les séquences des opérations d'usinage possibles en utilisant le réseau de neurones artificiel, basé sur l'algorithme d'entraînement par rétropropagation qui est pré-structuré avec la connaissance antérieure de domaine sous forme de règles heuristiques ou empiriques,

b) concevoir un format approprié pour la représentation des variables pour la prise de décision d'entrée et de sortie pour le réseau de neurones,

c) présenter une méthode systématique pour choisir les formes d'entrée pour les exemples à employer pour l'entraînement du réseau de neurones, ainsi qu'une méthode pour son entraînement et sa validation, et

d) accomplir la modification de la base de connaissance, si nécessaire.

Poursuivant les objectifs mentionnés ci-dessus, l'auteur a développé une méthodologie pour automatiser la tâche de la sélection des opérations d'usinage en utilisant une approche basée sur le réseau de neurones artificiels. Elle est capable de sélectionner automatiquement toutes les séquences alternatives des opérations possibles pour usiner chacun des types de particularités de la pièce. La méthodologie qu'on a proposée est applicable pour les pièces cylindriques usinées, qui sont symétriques par rapport à l'axe, et qui peuvent contenir les différents types de particularités tels les surfaces cylindriques externes, les surfaces frontales, les surfaces coniques, les filets, les rainures, les trous et les surfaces qui ne sont pas rotationnelles comme par exemple les gorges. Un progiciel intitulé NeuFrame est utilisé sur un micro-ordinateur pour simuler l'opération de réseau de neurones.

Le réseau de neurones a été d'abord pré-structuré avec la connaissance antérieure de domaine sous forme de règles heuristiques, qui ont été rassemblées de diverses sources telles que les livres et les manuels de l'usinage. Mais dans une situation pratique dans une entreprise, il est nécessaire de consulter aussi les catalogues appropriés des fournisseurs d'équipements de fabrication disponibles dans leur usine et les différents manuels de fabrication spécifiques à l'entreprise pour les détails sur des capacités des procédés de fabrication. Deux formes de représentation des données d'entrée au réseau de neurones ont été développées. La représentation externe est utilisée pour entrer les valeurs spécifiques des variables de la prise de décision (notamment le type des particularités de la pièce et ses attributs, comme par exemple le diamètre ou la largeur, la tolérance et la finition de la surface) au réseau neuronal. La représentation interne convertit les valeurs spécifiques des variables en des ensembles qui correspondent à diverses catégories de particularités géométriques et leurs attributs telles que représentées dans la partie « Si » des règles heuristiques mentionnées précédemment. La conversion des données d'entrée de son format de la représentation externe à la représentation interne est effectuée automatiquement en utilisant des règles de classification qui ont été développées dans ce travail de recherche. La couche d'entrée de neurones dans le réseau neuronal a été conçue de telle manière qu'un nœud neuronal soit assigné à chaque type de particularités et à l'ensemble de ses attributs. Dans la couche de sortie de réseau neuronal, un noeud est assigné à chacune des séquences réalisables parmi des opérations d'usinage qu'on retrouve dans la partie conséquente <<Alors>> des règles heuristiques. L'algorithme de rétropropagation a été employé comme mécanisme d'apprentissage pour entraîner le réseau neuronal. Une méthode systématique est illustrée pour choisir des formes d'entrée pour les exemples d'entraînement du réseau neuronal. Les règles heuristiques mentionnées précédemment ont été employées pour agir comme directives pendant la préparation des exemples d'entraînement de réseau neuronal; les formes d'entrée sont choisies de telle manière qu'elles activent une ou plusieurs de ces règles heuristiques. Ensuite, la méthode employée pour l'entraînement du réseau de neurones et leur validation est élaborée. On a aussi indiqué comment effectuer une modification de la base de connaissance s'il devient nécessaire. La modification de la base de connaissance peut être accomplie facilement et rapidement tout simplement par le réentraînement du réseau neuronal avec un nouvel ensemble d'exemples d'entraînement. Deux exemples des pièces cylindriques ont été analysés en utilisant l'approche de réseau de neurones proposée par l'auteur pour démontrer son potentiel d'application dans le vrai environnement de fabrication.

La méthode adoptée par l'auteur offre une approche générative de la génération automatique des gammes d'usinage pour les pièces complexes, qui peut aider à surmonter des limitations de l'approche variante. La méthode présentée n'est pas limitée uniquement aux pièces semblables ou pour lesquelles une gamme d'usinage existe déjà. D'ailleurs, elle peut générer les gammes d'usinage automatiquement et avec plus de consistance que la méthode variante. En éliminant le besoin de planificateurs experts pour la génération des gammes d'usinage, la méthode qu'on a proposée peut également réduire le temps de la génération des gammes d'usinage et, par conséquent, les coûts correspondants, et s'assure qu'il y ait moins d'erreurs humaines que dans l'approche variante. De plus, la modification de la base de connaissance peut être accomplie facilement par le réentraînement de réseau de neurones. L'approche qu'on a proposée offre aussi plusieurs avantages comparés à d'autres approches génératives comme par exemple l'arbre de décision et le système expert. Les avantages sont sa capacité d'acquérir automatiquement la connaissance, de généraliser et de générer les solutions aux problèmes où les données d'entrée contiennent des erreurs ou elles sont incomplètes. L'approche basée sur l'arbre de décision et le système expert sont relativement statiques en termes de la représentation de la connaissance. Mais dans l'approche qu'on a développée, la modification de la base de connaissance peut être accomplie facilement par le réentraînement de réseau de neurones. Enfin, par cette méthodologie basée sur le réseau de neurones, la tâche de la sélection des opérations d'usinage peut être effectué en investissant peu de temps, la rendant de ce fait très efficace et rentable pour des applications industrielles.

C. 4 Automatisation de la planification du montage par le système expert

Dans la dernière partie de ce travail de recherche, l'auteur traite du problème de l'automatisation des différentes tâches de la planification du montage, et la revue de la littérature montre que l'approche basée sur le système expert possède beaucoup de potentiel pour automatiser cette activité. Mais la plupart des efforts de recherche pour automatiser la tâche de la planification du montage en utilisant l'approche de système expert ont été limités au domaine des pièces prismatiques et leur potentiel pour les pièces cylindriques n'a pas été suffisamment exploré. De plus, dans la plupart des approches de système expert qui ont été rapportées par les autres chercheurs, un mélange de la méthode de système expert et d'une méthode algorithmique a été adopté, afin d'automatiser les différentes tâches de la planification du montage. Une limitation importante est que le système qui en résulte tend à être inflexible et répond mal aux nouvelles situations. En considérant tout ce qui est mentionné ci-dessus, les objectifs de la deuxième partie de ce travail de recherche sont définis comme suit:

a) développer une méthodologie de la planification du montage en utilisant le système expert pour faire le groupement des opérations d'usinage, l'ordonnancement des opérations et la sélection des surfaces de la pièce sur lesquelles il faut la localiser et la fixer,

b) concevoir un format approprié pour la représentation des données d'entrée pour le système expert,

c) développer l'ensemble de règles basées sur la connaissance pour les différents problèmes de planification du montage mentionnée ci-dessus,

d) accomplir la modification de la base de connaissance, si nécessaire.

Poursuivant ces objectifs, l'auteur a développé une méthodologie pour automatiser les différentes tâches de la planification du montage en utilisant une approche basée sur le système expert. Cette dernière est capable de faire automatiquement le groupement des opérations, l'ordonnancement des opérations dans chaque montage et de sélectionner des surfaces de la pièce sur lesquelles il faut la localiser et la fixer pour chaque montage. Elle a été implantée sur un micro-ordinateur en utilisant le progiciel CLIPS.

La base de données contient les données d'entrée pour le système expert et aussi des fonctions et des programmes externes qui sont nécessaires pour faire certains calculs. Les données d'entrée comprennent les détails de différentes particularités qui sont présentes dans la pièce comme par exemple, le type d'une particularité, ses dimensions, les relations de tolérance géométrique avec d'autres particularités, la direction d'accès d'outil pour usiner une certaine particularité et les opérations pour usiner chacune d'elles. Afin de représenter les données d'entrée mentionnées ci-dessus, un format de représentation a été conçu en utilisant le <<Template>>, qui est le format généralement utilisé pour la représentation de données d'entrée dans CLIPS. Il y a deux façons pour entrer les données au système expert qu'on a développé. Les données d'entrée peuvent être sauvegardées dans un fichier de données avec l'extension .clp et ensuite être chargées à partir de ce fichier dans l'environnement de système expert pendant son exécution. Alternativement, elles peuvent être également entrées manuellement entrée directement à partir d'une interface utilisateur.

La base de connaissance contient la connaissance du système expert. C'est la collection de règles qui est employée par le système expert pour les différents problèmes de planification du montage. Une base de connaissance comprenant les règles a été développée pour chacun des problèmes, notamment le groupement des opérations, l'ordonnancement des opérations et la sélection des surfaces de la pièce sur lesquelles il faut la localiser et la fixer pour chaque montage. La connaissance nécessaire pour formuler les règles mentionnées ci-dessus a été basée sur la connaissance heuristique et la connaissance experte de diverses sources telles que les livres et les manuels de l'usinage, et basée sur des entrevues avec les experts dans le domaine de la génération

des gammes d'usinage, aussi bien la discussion avec les machinistes experts que et les observations directes de l'usinage dans l'usine. Pour effectuer le groupement des opérations, un ensemble de règles a été développé, qui est capable de grouper les opérations d'usinage dans les différents montages. Les facteurs qu'on a considérés sont la direction d'accès d'outil pour usiner chacune des particularités et les relations de tolérances entres les particularités. Pour l'ordonnancement des opérations, un ensemble de règles a été développé, qui est capable de déterminer les séquences des opérations dans chaque montage selon les contraintes de priorité entres les différentes particularités ainsi que la logique de fabrication pour l'ordonnancement des opérations. Les contraintes de priorité entres les particularités mentionnées ci-dessus sont établies automatiquement par un ensemble de règles proposées par l'auteur. Pour l'ordonnancement des opérations, deux types de logique de fabrication ont été utilisés. Tout d'abord, les surfaces externes doivent être usinées et ensuite les surfaces internes. L'autre logique de fabrication consiste à usiner d'abord les surfaces brute, puis la demi-finition des surfaces et ensuite la finition. Un ensemble de règles, basé sur des principes heuristiques a été employé pour la sélection des surfaces de la pièce sur lesquelles il faut la localiser et la fixer pour chaque montage. Un des principes heuristiques mentionnés ci-dessus est de sélectionner une surface pour localiser et fixer la pièce pour un montage, s'il a une orientation différente des surfaces usinées dans le montage, et s'il a une relation de tolérance critique avec une des surfaces usinées. L'autre principe heuristique qu'on a utilisé est de sélectionner une surface pour localiser et fixer la pièce pour un montage, laquelle a le plus grand diamètre ou la plus longue surface cylindrique, et qui a une orientation différente des surfaces usinées dans le montage, quand il n'existe aucune relation de tolérance entres les surfaces de différentes orientations. Le moteur d'inférence à chaînage avant a été utilisé pour le système expert. La modification de la base de connaissance peut être faite en modifiant tout simplement les règles dans la base de connaissance du système expert. Les nouvelles connaissances peuvent être facilement acquises par les systèmes experts en introduisant de nouvelles règles à sa base de connaissance. Deux exemples des pièces cylindriques ont été analysés en utilisant

l'approche de système expert proposée par l'auteur pour démontrer son potentiel d'application.

La méthode adoptée par l'auteur offre une approche générative pour les différents problèmes de la planification du montage pour les pièces cylindriques. La nature modulaire de système expert et de la séparation du moteur d'inférence de la base de connaissance donne la flexibilité à l'approche proposée. La modification de la base de connaissance peut être accomplie facilement en modifiant simplement les règles dans la base de connaissance du système expert, ce qui prend moins de temps que la modification du programme original comme dans le cas des approches algorithmiques. De plus, les nouvelles connaissances peuvent être facilement acquises par l'introduction de nouvelles règles à la base de connaissance du système expert. Enfin, par cette méthodologie basée sur le système expert, la tâche de la planification du montage peut être effectuée en investissant peu de temps, la rendant très efficace et rentable pour des applications industrielles.

C. 5 Contributions et conclusions

Le travail de recherche rapporté par l'auteur dans cette thèse a apporté plusieurs contributions sur l'applicabilité des techniques d'intelligence artificielle, notamment le réseau de neurones et le système expert, afin d'automatiser les différentes tâches de la génération des gammes d'usinage pour les pièces cylindriques. Les contributions importantes sont récapitulées comme suit. Pour automatiser la tâche de la sélection des opérations d'usinage, une méthodologie intelligente basée sur le réseau de neurones, entraînée par l'algorithme de rétropropagation, a été développée. Elle tire profit de la connaissance antérieure de domaine (sous forme de règles heuristiques), qui aide à réduire la complexité de l'entraînement de réseau de neurones. Elle a été réalisée en développant un format de représentation des variables de la prise de décision pour les données d'entrée au réseau de neurones, par la pré-structuration de la couche d'entrée de réseau de neurones avec la connaissance antérieure de domaine. La sortie de réseau de neurones qu'on a développée est capable de générer automatiquement toutes les séquences des opérations possibles pour usiner une particularité. Chaque noeud de la couche de sortie du réseau représente une séquence des opérations d'usinage. Ceci permet au système d'explorer tous les plans alternatifs possibles afin de choisir le plan le plus approprié. Une méthode systématique a été illustrée pour choisir les formes d'entrées pour les exemples d'entraînement du réseau neuronal. Elle simplifie la préparation des exemples pour l'entraînement de réseau de neurones, aide à s'assurer que le domaine entier du problème est bien représenté, et exige peu d'exemples pour l'entraînement du réseau de neurones. L'implantation de la méthodologie pour la sélection des opérations d'usinage proposée est illustrée par un exemple d'une pièce cylindrique pour démontrer son potentiel d'application. Pour automatiser les différentes tâches de la planification de montage, une méthodologie intelligente basée sur le système expert a été développée en utilisant le progiciel CLIPS. Afin de représenter les données d'entrée au système expert, un format de la représentation a été conçu en utilisant le <<Template>>, qui est le format généralement utilisé pour la représentation de données d'entrée dans CLIPS. Un ensemble de règles a été développé pour chacun des différents problèmes de planification du montage, notamment le groupement des opérations en considérant la direction d'accès d'outil pour usiner chacun des types de particularités et les relations de tolérances entres les particularités, l'ordonnancement des opérations selon les contraintes de priorité entres les différentes particularités aussi bien que la logique de fabrication pour ordonnancement des opérations, et la sélection des surfaces pour localiser et fixer la pièce pour chaque montage basé sur des principes heuristiques. Les règles mentionnées ci-dessus ont été codifiées dans le langage de CLIPS pour implanter le système expert. L'implantation de la méthodologie pour la planification du montage qu'on a proposée est illustrée par deux exemples des pièces cylindriques pour démontrer son potentiel d'application.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

This research work aims at exploring possible applications of different Artificial Intelligence (AI) techniques for automating two of the important decision making tasks in generative Computer-Aided Process Planning (CAPP) systems, namely a) selection of machining operations and b) set-up planning, in the case of rotationally symmetrical parts. In this regard, an intelligent back-propagation neural network based methodology has been developed for the selection of all the possible alternative machining operation sequences. Also an intelligent expert systems based methodology has been developed for the different problems in set-up planning, such as the set-up formation, the operation sequencing and the selection of locating and clamping surfaces for each set-up.

1.1 Motivation, Research Issues and Problems

Process planning is an important manufacturing activity that interprets the part design specifications, and then systematically determines the detailed steps of manufacturing the part, according to the specifications and within the limitations of the available manufacturing resources and their technological capabilities. It thus acts as a bridge between design and manufacturing by translating the design specifications into manufacturing details (Chang and Wysk, 1985), and it is a major determinant of the manufacturing cost and quality of the part, manufacturing lead time and production efficiency (Allen 1987).

The process planning in discrete parts manufacturing by machining involves a number of decisions (Wang and Li, 1991) such as the design interpretation to determine the finished part requirements, the selection of raw stock, the selection of machining operations for producing each of the part surfaces, the selection of cutting tools and machine tools, the set-up planning which includes determination of set-ups for machining, determination of sequence of machining operations and method of locating and clamping the part for each set-up, the selection of the jigs and fixtures, the determination of cutting conditions, tool path planning, the generation of NC part programs and so on. The present research work focuses on the machining operations selection and the set-up planning. Traditionally in manually performed process planning, the process planner takes the above decisions, based on intuition and rules of thumb gained from his experience, and other skills such as his ability to interpret the part design and analyze the requirements for parts, his knowledge of machining processes and available manufacturing resources and their capabilities, and his ability to understand the interactions between the part, manufacturing, quality and cost. Manual process planning, however, has its own shortcomings (Chang and Wysk, 1985). Since it is a highly skilled job done by expert process planners, there is a high probability of losing all the skill along with the expert as he retires. Moreover, it is also laborious and time consuming and the resulting plans may be inconsistent due to variability in human process planning. To overcome these shortcomings, various Computer-Aided Process Planning (CAPP) approaches were developed. The main idea behind CAPP is to utilize the power of computers to help in automating the process planning tasks. The benefits (Wang and Li 1991) that can be obtained from CAPP are manifold and are as follows:

- It offers potential to capture the knowledge of the process planners into computer programs and thus, in effect, help to reduce the dependence on experts.
- It can significantly reduce the lead time and hence the cost for development of process plans.
- It can produce more accurate and optimal process plans.
- It can help to maintain more consistency in generation of process plans.
- It has the potential to provide an automated interface between the Computer-Aided Design (CAD) and Computer-Aided Manufacturing (CAM) systems and in the process help in achieving complete integration with the manufacturing

system, which is the ultimate goal of Computer Integrated Manufacturing (CIM) systems.

So obviously there is much interest among manufacturing organizations to implement the CAPP.

There have been numerous previous efforts to automate the process planning that have led to the development of different Computer-Aided Process Planning (CAPP) approaches (Chang and Wysk, 1985, Zhang and Alting 1994, Marri et al 1998). They broadly fall under one of the following categories, namely the variant CAPP approach, the generative CAPP approach or some hybrid of the two. The variant CAPP approach involves retrieval of an existing standard process plan for a similar part, followed by modification of the standard plan by an expert process planner. The generative CAPP approach, on the other hand, uses the knowledge of manufacturing and the decision making logic of process planning, captured and encoded into computer programs, to automatically create process plans for a new part from scratch without referring to existing plans. The generative CAPP approach has the following important advantages over the variant CAPP approach:

- The generative CAPP can be used to generate process plans for new parts with or without similarity to pre-existing parts.
- The generative CAPP has the potential to eliminate human intervention in planning process by automating the different decision making tasks involved.

The focus of the present research work concerns the generative CAPP approach. There are several key issues and challenges, which must be addressed for successfully developing and implementing any generative CAPP system. They include the following:

- An appropriate CAD/CAPP interface has to be developed for extraction of the part description knowledge from the CAD model with minimal human intervention. Also the extracted part description knowledge has to be represented in an appropriate form that is directly usable by the process planning system.

- Knowledge of manufacturing and the decision making logic of process planning have to be captured and encoded into appropriate computer programs that will be capable of generating the process plan automatically with minimal human intervention. An intelligent means of acquiring this knowledge with minimum human intervention has to be developed.
- In the case where several alternative process plans are possible for the same part, the process planning system has to be capable of optimising the process plans with respect to different factors such as cost and time
- Finally, the process planning system needs to be flexible i.e. capable of being modified easily so as to adapt to any major changes in the product design or the manufacturing environment such as manufacturing capability, capacity or availability of manufacturing resources, etc.

Keeping the above in mind, the focus of the present research work is aimed at developing intelligent methods for representing the manufacturing knowledge and the decision making logic of process planning for successful implementation of the generative CAPP systems. The traditional approaches such as algorithms and graphs, decision trees, and decision tables suffer from various shortcomings. In order for the algorithmic and graph theoretic to be successful, the program must contain all possible input-output combinations. In an extremely complex problem, the size of the program could become large and may need large computing resources. Moreover, they are inflexible, responding poorly to new situations. The main limitations with the use of decision trees and decision tables are that they are relatively static in terms of representing the process planning knowledge and they are inflexible. Furthermore, all of the above approaches lack the necessary intelligence to automatically acquire knowledge in the form of example data. To overcome some of these limitations, approaches based on the use of various Artificial Intelligence (AI) techniques (Monostori et al 1996, Meziane et al 2000) such as Expert System, Fuzzy Logic and Artificial Neural Networks

(ANN) were explored by researchers. The application of AI techniques to process planning leads to several benefits. They include the following:

- The application of expert system offers a number of advantages (Chang 1990) such as structured knowledge representation in the form of rules, an explicit inference route, explanation facility and ability to adapt to dynamic manufacturing environment by introduction of new rules.
- The application of fuzzy logic offers a number of advantages (Balazinski et al 1994, Ong et al 1997) such as a structured knowledge representation similar to that of expert systems in the form of rules with linguistic labels. Moreover, it is able to handle uncertainty and reason with imprecise information.
- The application of ANN offers various advantages (Wang and Li 1991, Monostori et al 1992)) such as capability to automatically acquire knowledge from examples, high processing speed, capability to adapt to changing environments through re-training, and ability to generalise and produce meaningful solutions to the problems where input data contains errors or is incomplete.

Several applications of the above approaches have been reported in literature. Since the present research work looks at the following two decision making tasks namely, the machining operations selection and the set-up planning, the focus here will be on these two only. The problems with existing methods to accomplish the above and the issues not adequately addressed by previous researchers are presented below.

Regarding solving the problem of machining operation selection, the literature review indicates that this problem has been solved by previous researchers using decision trees, and AI based techniques such as Expert Systems and ANN, and hybrid of the above approaches. Their research work shows that the application of ANN can overcome several limitations of the decision trees and the expert system based approaches. However, inspite of the above advantages, they have a set of problems of their own that includes, among others, choosing the set of training examples that is best representative of the problem domain. There are no guidelines at present on how to choose them and the values of input variables are often randomly selected from the entire range of values possible. Sometimes even after a large set of examples has been chosen, there may remain some cases where small sets of exceptions may be either unrepresented or poorly represented as a result of which such cases may be very difficult to handle correctly. Also an issue that has not been adequately addressed by previous researchers is that whether any prior domain knowledge, which is usually available on machining operations selection, could be taken advantage of. The prior domain knowledge is already well known to reduce the complexity of learning in ANN. Further most of the previously developed neural network models tend to recommend a single machining operation sequence for a given feature of the part.

As regards the problem of set-up planning, the literature review indicates that previous researchers have attempted to solve this problem using algorithmic and graph theoretic approaches, and various AI based techniques such as expert system, fuzzy logic, ANN and hybrid of the above. Their research work shows that the expert system has lot of potential for solving the problem of set-up planning. However, most of the research efforts so far have been limited to the prismatic parts domain. Their potential for application in set-up planning of rotational parts has yet to be fully explored. Also, in most of the expert system based approaches that have been reported thus far by previous researchers, a mixture of the expert systems and some algorithmic approach has been adopted in order to solve the set-up planning problems. One shortcoming of it is that the resulting system tends to be inflexible, responding poorly to new situations.

The research issues and challenges outlined above for successfully developing and implementing a generative CAPP system as well as the various problems with the existing methods and the issues identified above that have not been adequately addressed by previous researchers have motivated the author to undertake the present research work.

1.2 Objectives and Scope of the Research

The following objectives for this research work have been formulated:

Overall Objective

The overall objective is:

to explore possible applications of different AI techniques for automating two of the important decision making tasks in generative CAPP systems, namely a) machining operations selection and b) set-up planning in case of machined rotationally symmetrical parts

Specific Objectives

The specific objectives are:

1. i) to develop a back-propagation ANN based CAPP methodology for selection all the possible alternative machining operation sequences by prestructuring the neural network with prior domain knowledge in the form of heuristic or thumb rules,

   ii) to develop a suitable format of representation of input and output decision variables for the neural network,

   iii) to present a systematic method of choosing the input patterns for the examples to be used in training the neural network and a method for its training and validation, and

   iv) how to accomplish modification of the process planning knowledge base, when necessary
2. i) to develop an expert system based CAPP methodology for set-up planning to solve the problems of set-up formation, operations sequencing and selection of locating and clamping surfaces for each set-up,

   ii) to develop a suitable format for representation of the input data for the expert system,

iii) to develop a set of knowledge based rules for solving the various above set-up planning problems, and

iv) how to accomplish modification of the process planning knowledge base, when necessary.

The proposed CAPP methodologies are going to be capable of handling the presence of various rotational features that are commonly encountered in rotationally symmetrical parts.

1.3 Research Methodology

The neural network and expert system based methodologies developed for machining operations selection and set-up planning are briefly explained below.

For solving the problem of machining operations selection, a back propagation learning based feed forward neural network approach has been adopted by the author with the NeuFrame Version 4 (2000) software package being used to simulate the neural network operation. The neural network has been pre-structured with prior knowledge on machining operations selection in the form of heuristic or thumb rules. It has been achieved by developing two forms of representation for the input data to the neural network. The external representation is used to enter the crisp values of the input decision variables (namely the feature type and its attributes such as diameter or width, tolerance and surface finish) to the neural network. The purpose of internal representation is to categorise the above crisp values into certain sets; these sets, in turn, correspond to all the possible different ranges of dimension, tolerance and surface finish, encountered in the antecedent 'IF' part of the thumb rules mentioned above. The input layer of the neural network has been designed in such a way that one neuronal node is allocated for each of the feature type and the above sets of feature attributes. The output layer of the neural network has been designed in such a way that one neuronal node is allocated to each of the various feasible machining operation sequences found in the consequent 'THEN' part of the thumb rules. During the preparation of training examples, the thumb rules developed for selection of machining operation sequences have been used to serve as guidelines. The input patterns of training examples are chosen in such a way that they activate one or more of these thumb rules. It is done by selecting the input variable values that fall in the ranges related to those rules rather than randomly selecting the values of variables from their entire range as in the approaches adopted by previous researches. Any modification of the process planning knowledge base can be accomplished easily and quickly by merely retraining the network with a new set of training examples.

For solving the different set-up planning problems, a rule based expert system approach has been adopted by the author. It has been implemented on a PC by using the CLIPS rule-based expert system shell (Giarratano 1998). In order to represent the input data for the expert system, namely the features present in the part and the operations for machining each feature, a format for representation has been developed by the author using template that is the commonly used input data representation format in CLIPS. To carry out set-up formation, a set of rules has been developed, which are capable of clustering the machining operations into set-ups taking into consideration the Tool Access Direction (TAD) of the corresponding features and the relative tolerance relationships between them. Further, a set of rules has been proposed to establish the various feature precedence constraints and to determine the operation sequence in each set-up, subject to the above precedence constraints as well as manufacturing logic for ordering the operations. Also a set of rules, based on heuristic principles developed by the author has been used for selection of the locating and clamping features in each set-up. Any modification of the current set-up planning knowledge can be done by simply modifying the rules in the knowledge base of the expert system. Also new knowledge can be easily acquired by the expert systems through introduction of new rules to its knowledge base.

1.4 Organization of the Thesis

The thesis is organised as follows. In the remainder of this introductory chapter, some background on what is to follow in the next chapters has been briefly outlined. To start with, the various approaches to CAPP including the variant and generative approaches have been presented briefly. In regard to generative CAPP, the different part description methods and the different methods for representing the decision making logic including the decision tree, decision table, algorithmic and graph based approaches and various Artificial Intelligence based methods such as the expert system, neural network and fuzzy logic have been briefly presented. The general problem definition in machining process selection and set-up planning, which are the two process planning issues addressed by the author in the current research work, has been given.

In Chapter 2, a review of relevant literature has been presented on different generative process planning approaches developed by previous researchers to solve the problems of machining operation selection and set-up planning in CAPP systems. The reviewed research literature has been classified according the types of the generative CAPP approaches adopted and its applicability to the type of parts being planned. The relative advantages and limitations of the different approaches have been given.

In Chapter 3, the neural network methodology developed by the author for machining operations selection in rotationally symmetrical parts has been described. The detailed description of development of the neural network methodology including gathering of domain knowledge for formulating the thumb rules, topology of the neural network model, format of representation of the input and output decision variables for the network and the method of designing, training and validation of the network have been given. The potential for application of the developed neural network methodology has been illustrated with the help of some rotationally symmetrical parts and the results are presented. The major differences between the neural network methodology developed by the author and those adopted by earlier researchers have been pointed out along with some of the important advantages that stand to be gained from the approach proposed by the author.

In Chapter 4, the expert system based methodology developed by the author for set-up planning has been presented. The detailed description of development of the expert system methodology including the structure of the database and the knowledge base for solving the above set-up planning problems, and the inference engine have been presented. The examples of some industrially-relevant rotationally symmetrical workpieces have been analysed using the proposed approach to demonstrate its potential for application in the real manufacturing environment and the results are presented. The differences between the set-up planning approaches developed by the author and by previous researchers have been discussed along with their advantages of the developed approach.

Chapter 5 concludes the thesis by summarising the research and drawing conclusions, highlighting the major contributions, and by exploring scope for further work and future research directions.

1.5 Background

Some background on what is to follow in the next chapters is briefly outlined in this section. To start with, the various approaches to CAPP including the variant and generative approaches are presented briefly. In regard to generative CAPP, the different part description methods and the different methods for representing the decision making logic including the decision tree, decision table, and various Artificial Intelligence based methods such as the expert system, neural network and fuzzy logic are briefly described. The general problem definition in machining process selection and set-up planning, which are the two process planning issues addressed by the author in the current research work, is given.

1.5.1 Computer-Aided Process Planning (CAPP) Approaches

The following sections present briefly the various approaches to CAPP namely, the variant CAPP and the generative CAPP.

1.5.1.1 Variant CAPP Approach

The variant CAPP (also known as the retrieval CAPP) approach involves the use of computer as a tool to assist in identifying and retrieving existing process plan of a component that is similar to the component being planned. The process planner then edits the plan manually to create a "variant" of the existing plan so as to suit the specific requirements of the component being planned. The variant CAPP is implemented based on Group Technology (GT) and parts classification and coding. The steps in implementing the variant CAPP approach and how it is used to prepare process plans are briefly presented below.

The first step in implementing variant CAPP is to adopt an appropriate classification and coding system for the entire range of parts produced in the shop. An example of a commercially available classification and coding systems is the Optiz System (Optiz 1970). All the existing parts are then coded following the adopted scheme for coding. The next step is part family formation in which parts requiring similar processes are grouped into the same family. Before the part families can be formed, information concerning the processing of all existing parts has to be collected. Then techniques such as the Rank-Order Cluster Algorithm (King 1979) or the Direct Clustering (King and Nakornchai 1982) are used for grouping parts into families. Each family is then represented by a family matrix. The next step is to prepare a process plan, also called the standard process plan that can be used by the entire family. The standard process plans are then stored in a database and indexed by family matrices. After completion of the above steps, the variant CAPP system is ready for use. For a new part for which the process plan has to be determined, the first step is to derive the code of the part. The next step is to input the code to a part family search routine in order to find the family to which the part belongs. The search routine performs matching of the family matrix with a given code and returns the family number to which the part belongs. This family number is then used to identify and retrieve a standard plan. The standard process plan is examined to determine if any modifications are necessary. It may so happen that although the new part has the same code, there are still minor differences in the processes required to manufacture it. The process planner uses his experience to edit the standard plan accordingly. If a standard process plan does not exist for the given code, then a similar code for which a standard plan exists may be searched. Either by editing an existing standard plan or by starting from scratch, the process planner uses his experience to prepare the plan for the new part. The variant CAPP approach is schematically shown in Figure 1.1. Examples of some variant CAPP systems are CAPP of CAM-1 (Link 1976), Multi CAPP (OIR 1983), AUTOPLAN (Tempelhof 1979), MIPLAN (Schaffer 1980).

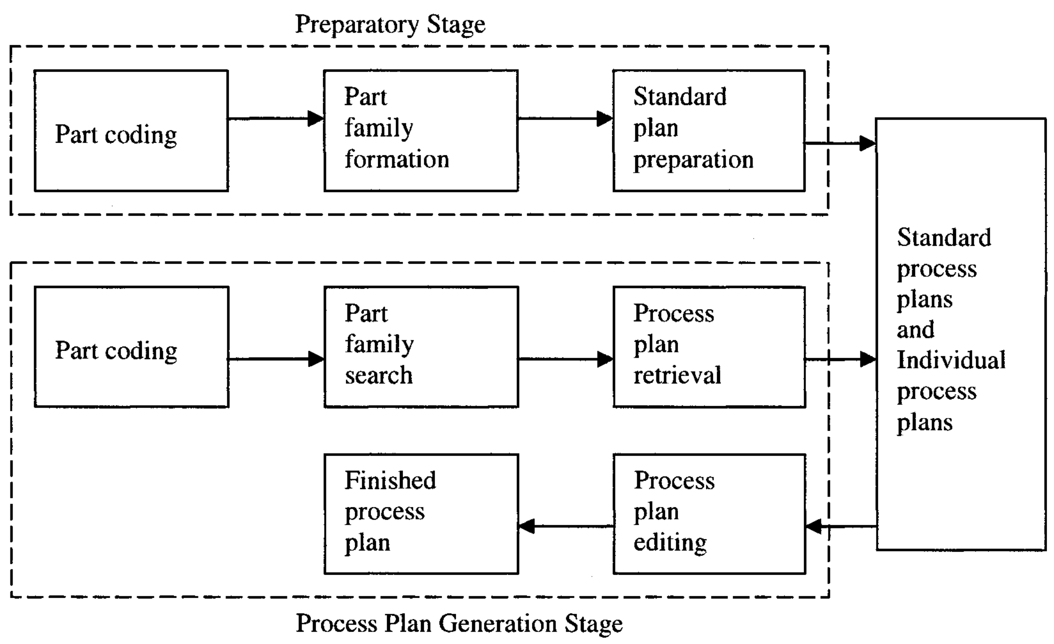


Figure 1.1. Variant CAPP Approach

1.5.1.2 Generative CAPP Approach

The generative CAPP approach represents an alternative approach to automated process planning. It uses the knowledge of manufacturing and the decision making logic of process planning, captured and encoded into computer programs, to automatically create process plans for a new part from scratch without referring to existing plans. A generative CAPP system comprises of three main components:

1) part description database
2) manufacturing databases and algorithms, and
3) decision making logic of process planning.

The part description database forms the input information needed for process planning. In addition to the part description database, various manufacturing databases e.g. process capability databases, machine tool databases, cutting tools databases, cutting parameters databases etc. are required to provide the process planning system with the information required to make decisions. Examples of various process planning functions and supporting databases for process planning are shown in Figure 1.2. Also algorithms need to be included in the process planning system for performing various computations and guiding the system during the decision making process. Possible applications of algorithms could be tolerance distribution, computation of process parameters which are, subject to solution by means of empirical formulas, determining the optimum cutting conditions and so on. The decision making logic of process planning includes the logic to be used by the process planner to make decisions on various aspects of process planning such as the process selection, the cutting tool and machine tool selection, the determination of process sequence, the set-up planning, the determination of cutting conditions and so on.

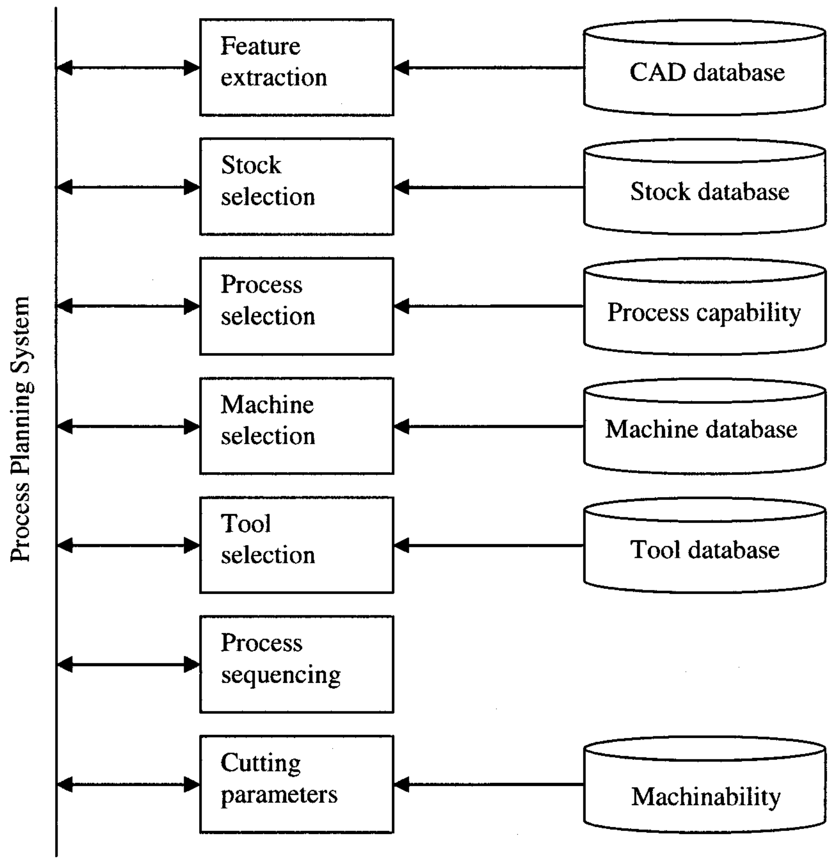


Figure 1.2. Examples of Some Process Planning Functions and Supporting Databases

1.5.2 Part Description Methods in Generative CAPP

Different methods have been used for representing the part description for generative CAPP. They include the following:

1) GT codes,
2) Special descriptive languages, and
3) 2-D and 3-D Computer-Aided Design (CAD) models.

Although GT codes are used commonly in variant CAPP systems, they have been used as well for part description in some generative CAPP systems such as M-GEPPS (Wang 1986 et al), CORE_CAPP (Li et al 1987). However, the codes used in generative CAPP systems need to be more detailed; often surface codes have to be specified describing the shape, dimensions, surface finish and tolerances of each surface in addition to the code describing the entire part. The degree of detail is limited by the resolution allowed by the number and type of digits used in the code. The code based representation is not suited for a completely automated generative process planning system, since the coding is a manual process, and a human interface between the design and the process planning functions is needed for translation of information from one system to another.

Special descriptive languages have been used for part description in generative CAPP systems such as AUTAP system (Eversheim 1980), GARI (Descotte and Latombe 1981). These languages are specially designed to contain all the information required for the necessary process planning functions. The majority of generative CAPP systems that use expert system approaches employ special descriptive language for input part description. Although the special descriptive language simplifies the process planning, it requires lot of effort from the user of the system who has to manually prepare the input data and is not suited for a completely automated generative process planning system.

Different 2-D and 3-D CAD models have been used and investigated for the input part description in generative CAPP systems such as CADCAM (Chang and Wysk 1985). The CAD model contains all the detailed information about a part design and so it has the capability to provide information necessary for planning functions. However, special feature recognition algorithms need to be used in order to extract information about the different machining features and their attributes that are necessary for performing the different process planning functions. The use of CAD models for input part description in generative CAPP systems has the potential to eliminate the human effort that is involved in translating a part design into GT code or special descriptive language formats.

1.5.3 Methods for Representing the Decision Making Logic in Generative CAPP

Several methods have been used to represent the decision making logic in generative CAPP. They include the following:

1) Decision tree,
2) Decision table,
3) Algorithmic and graph theoretic approaches, and
4) Various Artificial Intelligence (AI) based methods such as Expert Systems, Fuzzy Logic, Artificial Neural Networks (ANNs).

1.5.4 Decision Tree

A decision tree is a useful decision making tool. Its structure resembles that of a tree with a single root and a set of branches emanating from the root. The root is the start node and each branch emanating from the root can lead to another node or terminate in an action. There is a condition statement specified on each branch that must be satisfied in order to traverse that branch. If the condition specified on a branch is true, then that branch can be traversed to reach the next node, which, in turn, may provide further branching possibilities until a terminal point on the tree is reached. If the condition specified on a branch is true or false, then another branch might be taken. Figure 1.3 shows a sample decision tree. The algorithm for implementing a decision tree may be written in any of the procedural programming languages. An example of the application of decision tree for generative CAPP is the APPAS (Chang and Wysk 1985).

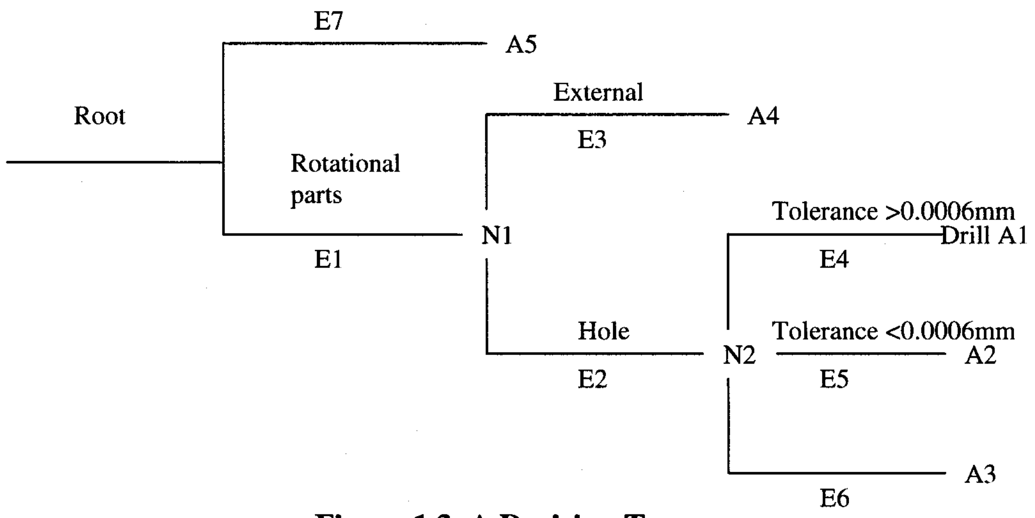


Figure 1.3. A Decision Tree

1.5.5 Decision Table

A decision table is another useful decision making tool. It organises the conditions, actions and decision rules in a tabular form. Conditions and actions are placed in rows, while decision rules are identified by columns. The upper part of the table includes the conditions that must be met in order for the actions represented in the lower part of the tables to be taken. When all the conditions in a decision table are met, a decision is taken. A sample decision table is shown in Figure 1.4. The algorithm for implementing the decision table may be written in either some specially developed decision table language or any of the procedural programming languages.

Condition	hole	X	X	X
	dia <3.0mm	X	X	X
	tol > 0.0006mm	X		
	0.02 < tol < 0.0006mm		X	
	tol < 0.0002mm			X
	sf > 0.005mm	X		
	sf < 0.005mm		X	X
Action	drill	1	1	1
	bore		2	
	ream			2

Note: dia indicates the diameter, tol indicates the tolerance and sf indicates the surface finish.

Figure 1.4. A Decision Table

1.5.6 Algorithmic and Graph Theoretic Approaches

Algorithms and graphs are powerful mathematical tools that have been used for solution of a large number of complex problems. An algorithm is a sequence of finite logical and mathematical expressions for the solution of a given well defined problem. Algorithmic approaches have been used in CAPP for set-up planning (Huang (1998). A graph is a collection of finite number of vertices and finite number of edges connecting a pair of vertices. A great number of problems can be formulated in terms of graphs. To solve, such problems, it is necessary to look at all the vertices or all the edges of a graph. The structure of the problem, for example, may be such that one only needs to traverse some of the vertices or edges, which is carried out by using an algorithm. Two common algorithms often used for traversing graphs are the depth-first-search and the breadth-first-search. Graph optimization algorithms are used to find the shortest path through the graph. Graph theoretic approaches have been used in CAPP for set-up planning (Huang et al 1996, Lee et al 2001and others).

1.5.7 Artificial Intelligence (AI) Based Methods

The different AI based methods for representing the decision making logic in generative CAPP are presented in the following sections.

1.5.7.1 Expert System

An expert system is a class of intelligent computer programs designed to solve problems in a specific domain of application, which would otherwise require significant human expertise for their solution, by using domain specific knowledge and inference procedures. The major parts of an expert system are:

a) Declarative knowledge about the problem

b) Problem solving knowledge or domain knowledge or procedural knowledge,

c) Inference engine, and

d) User Interface, Explanation Facility, Knowledge Acquisition Facility.

Figure 1.5 shows the block diagram of an expert system.

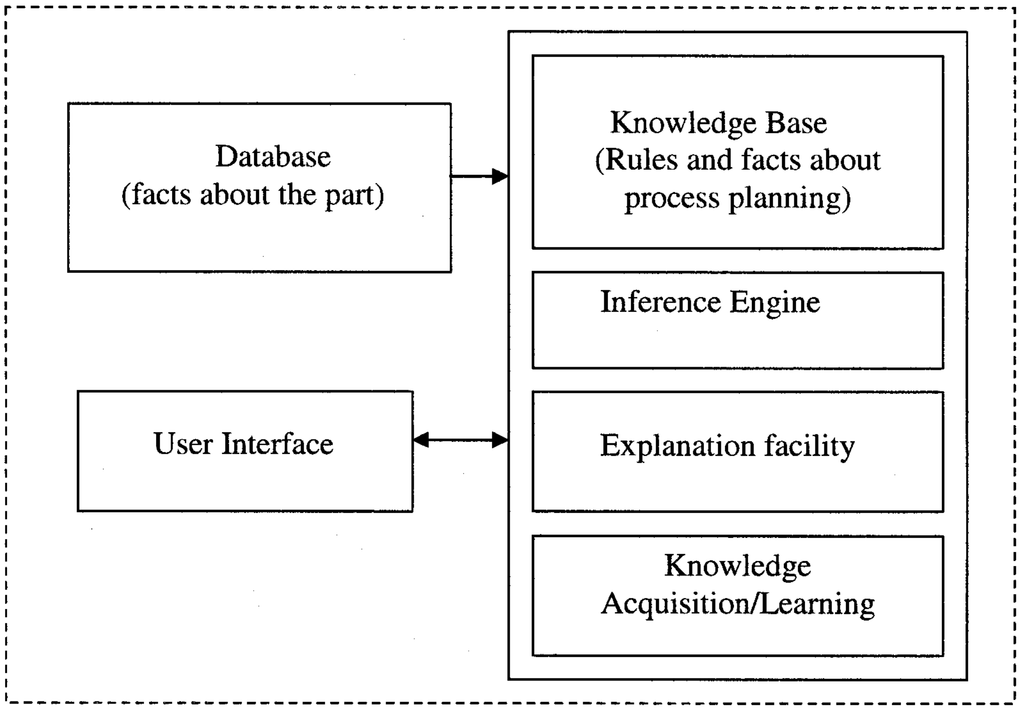


Figure 1.5. Block Diagram of an Expert System

a) Declarative Knowledge

The declarative knowledge constitutes the input to the expert system. It consists of facts about the problem being solved, which, in the case of process planning, include detailed information about the part being planned, as contained in the original part design on a CAD system. The part information necessary for process planning may include geometry, geometric relationships, dimensions and tolerances, and additional manufacturing specifications. The above information needs to be converted from its original data format on the CAD system into the representation format for declarative knowledge of the expert system. Some common forms of representation of the declarative knowledge facts include frames, feature based representation and first order predicate calculus and so on.

b) Procedural Knowledge

The problem solving knowledge or domain knowledge or procedural knowledge of the expert system is specific to a problem domain, which in the present case, is process planning; it consists of facts and IF-THEN rules about process planning. The IF portion (also called antecedent) of the rule is a series of patterns, which specify the facts, which cause the rule to be applicable. The THEN portion (also called consequent) of the rule is a list of actions to be executed when the rule fires.

c) Inference Engine

The third major part of the expert system is the inference engine that contains the general problem solving knowledge. It determines which rule antecedent, if any, is satisfied by the facts. Two general methods of inferencing are commonly used in the inference engine namely,

- Forward chaining
- Backward chaining.

With the forward chaining or data-driven inferencing, the system tries to match the available facts with the antecedent part of the rule. When matching rules are found, one of them is fired i.e. the consequent part of the rule is executed, generating new facts, which in turn cause other rules to fire. This process continues until no more applicable rules are found. In backward chaining or goal driven inferencing, a goal to be proved is specified. If the goal cannot be immediately satisfied by existing facts, the system will examine all the rules in the knowledge base for particular rules with the goal in their consequent part. Next the system will determine whether there are facts that will cause any of those rules to fire. If such facts are not available, they are set up as subgoals. This process continues recursively until either all the required facts are found and the goal is proved or any of the subgoals cannot be satisfied, in which case, the original goal is disproved. The choice of a particular inferencing method depends on the particular problem.

d) User Interface, Explanation Facility, Knowledge Acquisition Facility

In addition to the above three major components, an expert system may have three other components namely, a user interface, an explanation facility and a knowledge acquisition module. The user interface provides the mechanism by which the user can communicate with the expert system. The explanation facility displays the rationale behind applying (firing) a certain rule and is helpful in debugging the knowledge base. The knowledge acquisition module assists the user at the time of updating the knowledge base. Whenever new rules are added to the knowledge base, it helps to ensure that the new rules are consistent with the existing rules and if there is any contradiction of the newly entered rule with the existing rule, it is detected by this module.

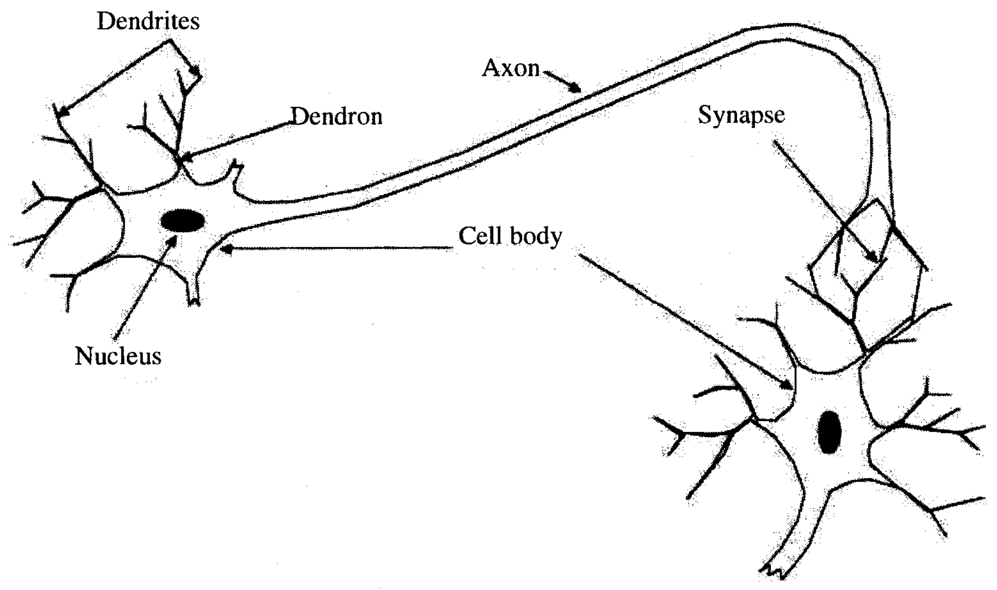
There are many tools, which have been used to build expert systems. These include Artificial Intelligence languages such as LISP and PROLOG and programming languages such as FORTRAN and C. However writing a complete expert system using any of the languages requires a tremendous amount of work. Therefore, to ease the task of building an expert system, several expert system shells and expert system building tools are available. Expert system shell is a special purpose tool designed for certain types of application in which the user must supply only the knowledge base. Examples of expert system shells are EMYCIN and CLIPS. Using expert system shells can speed up the expert systems development time. However, shells were originally developed for other applications; therefore, for a different application it must be fitted into the existing shell for which some compromise may have to be made. Expert system building tools, although similar in features to an expert system shell, provide the users with a tool to build their own application. Examples of such tools are KEE, KnowledgeCraft.

The expert systems have been used in CAPP applications to automate a wide range of process planning tasks that include process selection, machine selection, cutting tool selection, cutting conditions selection, process sequencing and determination of set-ups. Examples of application of generative CAPP systems developed using the expert system are the EXCAP (Davies et al 1984), SIPP (Nau and Chang 1985).

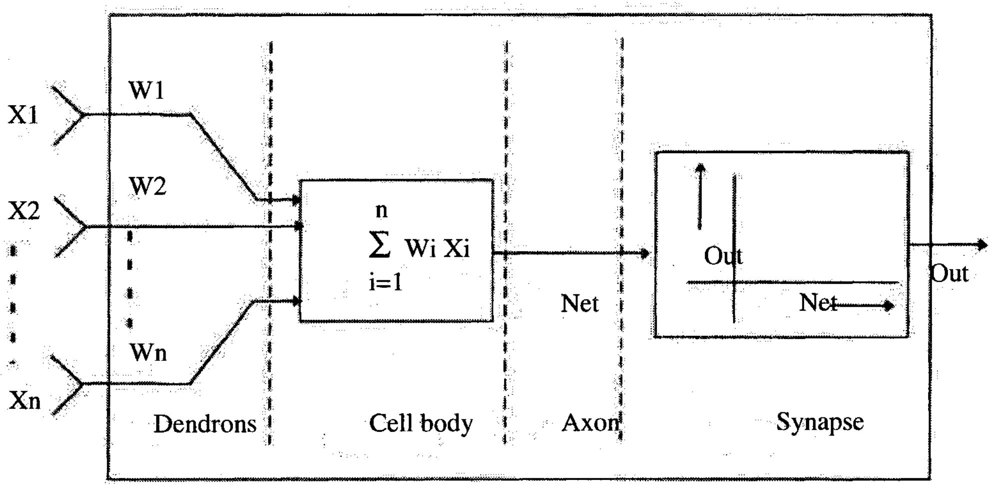
1.5.7.2 Neural Networks

Artificial neural networks are a class of computing systems that is loosely modelled after biological neural networks found in the human brain and nervous systems, and it uses a highly parallel architecture to efficiently perform various pattern recognition, pattern prediction and pattern classification tasks.

An artificial neural network is designed to act fundamentally like a biological neural network whereas in biological neurons, the dendrite receives electrical signals from the axons of other neurons; in the artificial neuron, these electrical signals are represented as numerical values. In a biological neural network, at the synapses between the axons and dendrites of the neighbourhood neurons as shown in Figure 1.6a, the electrical signals are modulated in various amounts. This is also modelled in the artificial neuron by multiplying each input value by a value called the weight. A biological neuron fires an output signal only when the total strength of the input signals exceeds a certain threshold. This phenomenon modelled in an artificial neuron by calculating the weighted sum of the inputs to represent the total strength of the input signals, and applying a suitable activation (threshold) function on the sum to determine its output. As in biological neural networks, this output is fed to other artificial neurons. The schematic diagrams of a biological neuron and an artificial neuron are shown in Figure 1.6.



(a) A Biological Neuron



(b) An Artificial Neuron

Figure 1.6. Biological Neuron and Artificial Neuron

The artificial neural network models are specified mainly by their

a) topology, and

b) method of training.

a) Topology of Neural Networks

The neural networks may be organized into different structural arrangements (topologies) according to the nature of arrangement of neurons and the connection patterns of the layers. The important neural network topologies include:

- feed forward neural networks,
- feedback neural networks, and
- self-organizing neural networks.

Of the above, feedforward neural networks are the most popular and most widely used of the neural network topologies. They are known by many different names, such as "multi-layer perceptrons". Figure 1.7 shows a feed forward architecture of the neural network. Each layer contains a number of neurons, depicted by circles in the figure. Each layer has full interconnection to the next layer, but no connections within a layer. The first layer of the network is known as the input layer, whose neurons take on the values corresponding to the different variables representing the input pattern. The second layer is referred to as a hidden layer, because its outputs are used internally and not considered as output of the network. There may be more than one hidden layer present (In Figure 4.7, there are two hidden layers). The final layer of the network is known as the output layer. The values of the neurons of the output layer constitute the response of the neural network to an input pattern presented at the input layer. The number of layers and the number of neurons in each hidden layer are among the important user design parameters to be considered at the time of construction of the neural network model. The general rule is to choose these design parameters so that the best possible model with as few parameters as possible is obtained. The two other neural network topologies include the feedback neural networks and self-organizing neural networks.

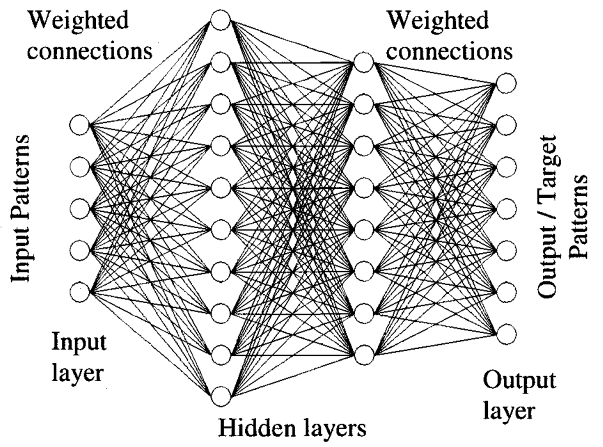


Figure 1.7. General Architecture of the Neural Network

b) Methods of Training of the Neural Networks

The usefulness of an artificial neural network comes from its ability to respond to input patterns in some orderly (desirable) fashion. For this to occur, it is necessary to first train the network to respond correctly to a given input pattern. This process is called training. It is an iterative process that adjusts the parameters (weights) of the neural network until the network is able to produce the desired output from a set of inputs. The process of training a neural network can be broadly classified into mainly two categories namely,

- supervised learning, and
- unsupervised learning.

A number of training algorithms based on the supervised learning are available, of which the most common is the backpropagation algorithm. The process of training a neural network using this algorithm is briefly presented below. The backpropagation algorithm first of all supplies the neural network with a sequence of input patterns and desired output (target) patterns, which together constitute what is known as the training examples. As an input pattern is presented to the neural network, the output response is calculated on a forward pass through the network. Then this output is compared to the desired output and error terms are calculated for each output neuron. The error is then fed backwards through the network and weights of each of the interconnected neurons are adjusted in such a way that the error between the desired output and the actual output is reduced. The propagation of errors back through the network gives backpropagation its name. The process of supervised learning using backpropagation algorithm is schematically shown in Figure 1.8.

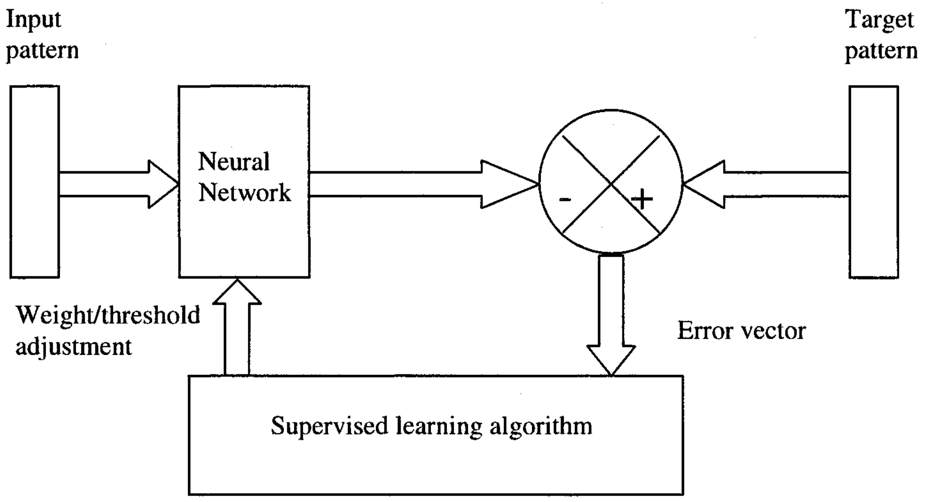


Figure 1.8. The Supervised Learning Process

The process of training the neural network using supervised learning is applicable to problems, where representative examples of both input pattern and output (target) patterns are known. However in many problems, where the target patterns are unknown, the unsupervised learning process is used. In the unsupervised learning process, the network is provided with a data set containing input patterns but not with desired output patterns. The unsupervised learning algorithm then performs clustering of the data into similar groups based on the measured attributes or features of the given input patterns serving as inputs to the algorithms. Loosely speaking, supervised learning is analogous to a student guided by an instructor, while unsupervised learning is analogous to a student who derives the lesson totally on his or her own.

Neural networks have been used in CAPP systems for automating various process planning tasks such as operation selection (Knapp et al 1992), selection of parameters of cutting tools (Santochi et al 1996), etc.

1.5.7.3 Fuzzy Logic

Fuzzy logic based systems are a class of computing systems that allow modelling of the human imprecise reasoning process with fuzzy sets. They resemble human decision making in its ability to draw definite conclusions from vague, ambiguous or imprecise information.

They have been designed to overcome a major disadvantage of rule-based expert systems, that of the inability to handle new situations not covered explicitly in their knowledge bases (that is, situations not fitting exactly those described in the antecedent part of the rules). With fuzzy logic, the precise value of a variable is replaced by a linguistic description, the meaning of which is represented by a fuzzy set, and inferencing is carried out based on this representation. Fuzzy set theory may be considered an extension of classical set theory. While classical set theory is about 'crisp' sets with sharply defined boundaries, fuzzy set theory is concerned with 'fuzzy' sets whose boundaries are 'blurred' or 'fuzzy'. The benefit of replacing the sharply defined boundaries with the so called fuzzy boundaries is the strength in solving real-world problems, which certainly involve some degree of imprecision and noise in the variables and parameters measured and processed for the application.

A fuzzy logic based system is similar in many features to that of an expert system, i.e. it consists of

a) declarative knowledge about the problem
b) problem solving knowledge or domain knowledge or procedural knowledge, and
c) inference engine.

However, the procedural knowledge in fuzzy logic based systems is expressed as fuzzy IF.. THEN... rules and the fuzzy inference engine is designed to carry out inferencing based on this representation. In addition to these three major components, the fuzzy logic based system incorporates two others components for

- Fuzzification of terms that appear in the antecedent part of the rules, and
- Defuzzification of fuzzy terms that appear in the consequent part of the rules into crisp values.

Fuzzy logic systems have been used in CAPP systems for automating various process planning functions such as set-up planning (Ong et al 1997), selection of cutting parameters for machining (Balazinski et al 1994, Hashmi et al 1998), etc.

1.6 General Problem Definition in Machining Process Selection and Set-up Planning

The selection of machining operation sequences is that part of the process planning task in which a set of operation sequences are selected for machining the part. It is an important activity of process planning, since it determines the nature and direction of other process planning decisions such as selection of cutting tools, machine tools, set-ups, cutting conditions and so on. Usually in order to machine a part, first of all, a suitable operation sequence has to be selected for machining each feature of the part. The main factors to be considered in selecting a machining operation sequence are various attributes of the feature such as dimensional size (e.g. diameter, length, width, etc.) of the feature, the tolerance and the surface finish specifications. Based on the particular values of a feature's attributes, it is possible to identify a sequence of machining operations to produce that feature. In order to select machining operation sequences, it is necessary to have prior knowledge about the various machining methods available in the shop floor and their capabilities in terms of the attainable dimension, tolerance and surface finish. The sequence of operations for machining each feature is selected by matching the process capabilities of the available machining methods with the dimension, tolerance and surface finish requirements of feature to be machined.

Set-up planning involves mainly the following three tasks: the set-up formation i.e. determination of groups of machining operations that can be carried out in the same set-up, the determination of sequence of operations within a set-up and the selection of the set-up datums. The decision on set-up formation is made considering the tool access directions (TADs) (either left or right) of the features present in the part to be machined and geometric tolerance relationship (if any) between features. For a given machine tool, the operations on features with the same TAD are grouped in the same set-up. Features having more than one TAD are assigned a unique TAD, depending on their tightest tolerance relationship with other features, so that their corresponding machining operations can be grouped along with other operations in one of the set-ups. The decision on determining the sequence of operations is based in accordance with the constraints imposed by the precedence relationships between features and machining operations to produce them, and the manufacturing logic to be followed in ordering the machining operations of various features such as for example, first machining of external surfaces, followed by machining of internal surfaces and first rough machining, followed by semi-finish machining, followed by finish machining and so on. Finally the selection of datum features for locating and clamping is performed with the purpose of obtaining the remaining critical tolerance relationships between features that could not be satisfied during the set-up formation and leaving the inevitable tolerance chain errors to those unimportant relationships with looser tolerances. It is accomplished with a set of heuristic principles for datum selection. For each set-up, a cylindrical and a vertical surface are needed as datums to locate and clamp the part. Therefore, there are actually four surfaces, two cylindrical and two vertical, to be used for the two set-ups. For the four surfaces, the two on the right are to be used to machine the surfaces on the left and the two on the left to machine the surfaces on the right. The approach for solving the different set-up planning problems is shown in Figure 1.9.

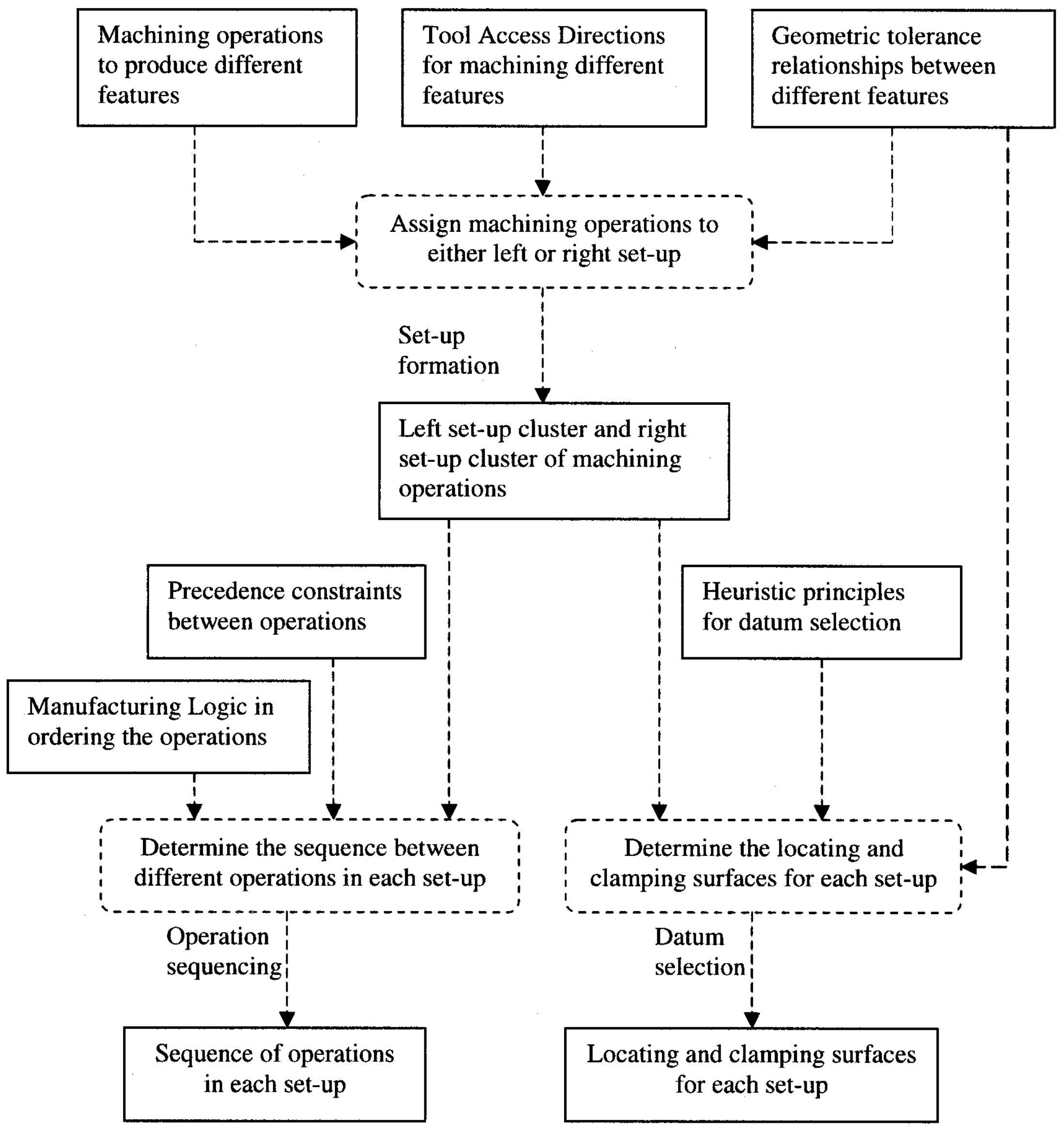

Machining operations to produce different features
Tool Access Directions for machining different features
Geometric tolerance relationships between different features
Assign machining operations to either left or right set-up
Set-up formation
Left set-up cluster and right set-up cluster of machining operations
Precedence constraints between operations
Heuristic principles for datum selection
Manufacturing Logic in ordering the operations
Determine the sequence between different operations in each set-up
Determine the locating and clamping surfaces for each set-up
Operation sequencing
Datum selection
Sequence of operations in each set-up
Locating and clamping surfaces for each set-up

Figure 1.9 Approach for solving the different set-up planning problems

CHAPTER 2

LITERATURE REVIEW

A number of generative process planning approaches have been developed by previous researchers to solve the problem of selection of machining operation sequences and that of set-up planning in CAPP systems. Some selected research articles, which address the above two problems in CAPP will be reviewed here. They have been classified, according the type of generative process planning approach adopted, into the following: approach based on decision trees, algorithmic and graph based approach, various Artificial Intelligence (AI) based approaches (such as the expert systems based approach, neural network based approach, fuzzy logic based approach) and hybrid AI based approach, which may be a combination of any of the above AI based approaches. Table 2.1 summarises the reviewed research literature based on the type of generative process planning approach adopted and its applicability to the type of parts being planned.

Table 2.1. Summary of the Reviewed Research Literature on Selection of Machining Operation Sequences and Set-Up Planning

Nature of the problem in CAPP	**Generative process planning approach adopted**	**Applicability to the type of parts being planned / type of features**	**Researchers**
Set-up planning	Expert System	Prismatic parts	Joshi et al (1988)
Set-up planning	Expert System	Prismatic parts	Wang and Wysk (1988)

Table 2.1. (Continued) Summary of the Reviewed Research Literature on Selection of Machining Operation Sequences and Set-Up Planning

Nature of the problem in CAPP	**Generative process planning approach adopted**	**Applicability to the type of parts being planned / type of features**	**Researchers**
Set-up planning	Expert System	Prismatic parts	Chang (1990)
Selection of machining operation sequences	Supervised neural network	Rotational parts	Knapp and Wang (1992)
Set-up planning	Unsupervised neural network	Prismatic parts	Chen and Steven LeClair (1993)
Selection of machining operation sequences	Expert system	Parts in general and Features such as holes	Khoshnevis and Tan (1995)
Selection of machining operation sequences	Expert system	Prismatic parts and Features such as holes, slots, pockets and curved surfaces	Wong and Siu (1995)
Datum selection	Neural network	Rotational parts	Mei, Zhang and Oldham (1995)

Table 2.1. (Continued) Summary of the Reviewed Research Literature on Selection of Machining Operation Sequences and Set-Up Planning

Nature of the problem in CAPP	**Generative process planning approach adopted**	**Applicability to the type of parts being planned / type of features**	**Researchers**
Selection of machining operation sequences, Set-up planning	Expert system, Constraint programming	Prismatic parts and Features such as plain, round, threaded holes, bored holes, face milled flat surfaces and simple pockets	Sabourin and Villeneuve (1996)
Selection of machining operation sequences	Hybrid AI (Neural network-fuzzy logic)	Parts in general and Features such as holes	Huang, Zhang and Sun Shan (1996)
Set-up planning	Graph based approach	Rotational parts	Huang and Zhang (1996)
Set-up planning	Fuzzy logic	Prismatic parts and castings	Ong and Nee (1997)
Set-up planning	Hybrid AI (Hopfield network-Simulated Annealing)	Prismatic parts	Chen, Zhang and Nee (1998)

Table 2.1. (Continued) Summary of the Reviewed Research Literature on Selection of Machining Operation Sequences and Set-Up Planning

Nature of the problem in CAPP	**Generative process planning approach adopted**	**Applicability to the type of parts being planned / type of features**	**Researchers**
Set-up planning	Expert systems, Mathematical programming	Prismatic parts	Kim et al (1998)
Selection of machining operation sequences	Expert system	Prismatic parts	Jiang, Lau, Chang and Jiang (1999)
Selection of machining operation sequences	Decision trees	Rotational parts	Wang (1998)
Set-up planning	Algorithmic approach	Rotational parts	Huang (1998)
Selection of machining operation sequences	Supervised neural network	Rotational parts	Devireddy and Ghosh (1999)

Table 2.1. (Continued) Summary of the Reviewed Research Literature on Selection of Machining Operation Sequences and Set-Up Planning

Nature of the problem in CAPP	**Generative process planning approach adopted**	**Applicability to the type of parts being planned / type of features**	**Researchers**
Selection of machining operation sequences	Hybrid AI (Neuro-fuzzy network)	Rotational parts	Maiyo, Xiankui, Chengying (1999)
Set-up planning	Graph based approach	Prismatic parts	Zhang et al (1999)
Set-up planning	Self-organising neural networks, Hopfield neural networks	Prismatic parts	Ming and Mak (2000)
Selection of machining operation sequences	Expert system	Parts in general and Surfaces of the types: flat, contoured, rotational, spherical, and features such as holes	Radwan (2000)
Set-up planning	Hopfield neural network	Prismatic parts	Chang and Angkasith (2001)
Set-up planning	Graph based approach	Rotational parts	Lee at al (2001)

Table 2.1. (Continued) Summary of the Reviewed Research Literature on Selection of Machining Operation Sequences and Set-Up Planning

Nature of the problem in CAPP	**Generative process planning approach adopted**	**Applicability to the type of parts being planned / type of features**	**Researchers**
Selection and sequencing of machining operation sequences	Supervised neural network	Rotational parts	Devireddy, Eid and Ghosh (2002)
Set-up planning	Graph based approach	Rotational parts	Shunmugam, Mahesh and Reddy (2002)

2.1 Different Approaches for Selection of Machining Operation Sequences in CAPP

In this section, details of the different generative process planning approaches listed in Table 2.1 for solving the problem of selection of machining operation sequences in CAPP will be briefly discussed along with their advantages and limitations.

2.1.1 Decision Tree Based Approach

Wang (1998) developed a decision tree based approach for process selection in machined rotationally symmetric parts. The decision trees are used to classify the profiles encountered in the part into lines and/or arcs, which are, in turn, classified further into vertical, horizontal or slant in case of lines and convex or concave in case of arcs and according to their directions, either positive or negative. This classification helps to identify the shape of the area to be removed from the raw material bar stock after analysing the sequence of the lines or arcs encountered. Then depending on the shape, each area is associated with a material removal operation as for example, turning is considered, if a rectangular area is to be removed.

The main limitation with the use of decision trees is that they are relatively static in terms of representing the process planning knowledge. Since this knowledge is usually coded line by line in a program, so any modification of the current knowledge would require rewriting of at least some portion of the original program. Furthermore, they lack the ability to automatically acquire knowledge in the form of example data.

2.1.2 Expert Systems Based Approach

Different expert system based approaches have been used by previous researchers for solving the machining process selection problem. These include the use of forward planning and backward planning approaches of expert systems, different tools for building expert systems such as AI programming languages like Prolog, LE-LISP, Expert System shells like Knowledge Craft and Expert System building tools like ART-IM. The following gives details of these approaches.

Khoshnevis et al. (1995) have used an expert system based approach for process selection for hole making processes. It has been implemented using an Expert Systems Shell called Knowledge Craft. The manufacturing process capabilities of different hole making processes such as dimensions, dimensional and geometric tolerances and surface finish capabilities have been stored using the frame representation. Two sets of rules have been developed for selection of basic processes and for further break up of the basic processes into rough, semi finish and finish operations. These rules are used to generate all possible operation sequences for a given hole by matching its surface requirements with the capabilities for each process. The method of process selection uses a forward reasoning based inference mechanism that begins with the initial state (raw stock) and then searches forward through the rule base to find operations in order to reach the final state of the part. The input data for process selection can be either entered manually by the user or obtained from a feature recogniser that is capable of automatically identifying the manufacturing features from a CAD solid model of the part.

Wong et al (1995) have used an expert system based approach to machining process selection for complex prismatic parts containing different kinds of holes, slots, pockets and curved surfaces. The input data for process selection that includes the geometric and technological information, such as surface finish and tolerance, of all machined surfaces is obtained directly from the CAD system, where the part is modelled using machining features and represented by a surface tree structure denoting the surface stacking relationships. For carrying out process selection and sequencing, several algorithms have been developed and implemented using Prolog. An algorithm is first used to generate the preliminary operation precedence. It uses rules from the process capability knowledge to assign a machining process to each machined surface and rules from the surface priority knowledge to determine the sequence of machining the surfaces. This algorithm essentially transforms the surface tree structure of the part description from the surface stacking relationship to an operation precedence. Basically this algorithm is based on a backward planning procedure as it starts from a finished part and tries to fill it back to the blank state. After generation of the preliminary precedence, it is refined further using an algorithm using refinement knowledge, such as drilling operations may be refined to further include centre drilling, pilot drilling, etc. Finally, the refined operation precedence represented as a tree structure is linearized into the final required operation sequence by using a rule based algorithm that groups together the machining operations using the same machine tools and / or cutting tools so that the unnecessary part loading / unloading on machine tools and cutting tool changes are eliminated, while at the same time the precedence relationship between operations is maintained.

Sabourin et al (1996) have used an expert system based approach for generation of machining operation sequences for parts containing features such as plain, round, threaded holes, bored holes, face milled flat surfaces and simple pockets. The expertise for machining operations selection for different types of features has been gathered from the planning specialists and a knowledge base has been developed using rules for automatic generation of machining operation sequences. For each feature, the rules are used to generate the machining operation sequence in a step-by-step manner by successively adding machining operations to the operation sequence, as and when a certain machining operation is necessary, based upon the analysis of different attributes of the feature to be machined. The dimensions, tolerance and surface finish of each feature are considered among other factors for machining operations selection. The knowledge base has been developed with the LE-LISP language. The input data necessary for operations selection is obtained directly from the CAD system CATIA, where the part is modelled using machining features.

Jiang et al. (1999) have used an expert system based approach to process selection for machined prismatic parts. It has been implemented by a rule based expert system building tool, ART-IM. The machined shape-producing capability knowledge has been represented by IF-THEN rules. These rules together constitute the knowledge base, which is used to decide the processes and sequence of operations needed to produce different types of machined shapes such as plane surface, step, slot, pocket, contour, holes, etc. The inputs to the process selection module include type of each machined shape along with the dimensions, tolerance and surface finish of each machined shape. This information is first manually entered by the user according to the component design description or engineering drawings and then it is automatically converted into a format that is directly usable by the process selection module.

Radwan (2000) has proposed an expert system based approach to process selection for CAPP systems based on a relational model. It is applicable to parts with flat surfaces, contoured surfaces, rotational surfaces, holes and special surfaces. Process diagrams and flow charts have been developed for each of these different types of surfaces. They relate the manufacturing processes to the various parameters of the surface, such as material type, dimensional restrictions, dimensional tolerances and surface finish that influence process selection. A relational model has been formulated between the surface characteristics and manufacturing processes based on the above process diagrams and flow charts. This model is used to perform the process selection by matching the given surface characteristics with those from the process diagrams and flow charts. The relational model has been implemented using SQL. The input data for process selection such as different surface characteristics can be entered either manually or by some automatic means from the CAD system.

The application of knowledge based expert systems offers number of advantages. It offers a structured knowledge representation in the form of rules that is easily understandable and editable by the user. The modular nature of expert systems makes them easier to encapsulate the knowledge and expand them by incremental development. Separation of control knowledge or inference engine from the knowledge base gives added flexibility to the expert systems. By keeping track of which rules have been fired, the explanation facility of the expert system can present the chain of reasoning that led to a certain conclusion. This gives added confidence to the system. The expert system is able to adapt to the changing manufacturing environment by its ability to acquire new knowledge through introduction of new rules to its knowledge base.

The expert systems, however, suffer from some weaknesses. Firstly, it is restricted to the fields where expert knowledge is available and it is unable to infer when information provided is incomplete. Further the expert system is known to perform exhaustive searches of its knowledge base for matching the patterns resulting in increased execution times with increase in number of rules. It is unable to automatically acquire the inference rules. The new knowledge must be built into the expert systems by specifying the new knowledge in explicit rule format. Furthermore, any modification of the knowledge base can be a tedious job in certain situations as it involves not just the introduction of the new rules, but also conflict resolution with the pre-existing rules.

2.1.3 Neural Networks Based Approach

Different approaches based on the supervised neural network that uses back-propagation training algorithm have been used to solve the machining process selection problem. The following gives details of these different neural network based approaches.

Knapp et al (1992) has demonstrated the ability of neural network in the process selection and within-the-feature process sequencing. In this work, two co-operating neural networks have been utilised. The first one, a three layer back propagation neural network, takes in as input the attributes of a feature and proposes a set of machining alternatives. Then another fixed weight neural network selects exactly one of the alternatives. Parameters of the features are modified by the results of the operation until the final state of the feature has been reached.

Devireddy et al (1999) has used a back propagation neural network based approach for selection of manufacturing operations for rotational components containing machining features like hole, step, taper and thread. Two stages of decision making have been implemented in the manufacturing operations selection approach. The first stage identifies the basic manufacturing operations needed to generate the features and their sequences by a neural network. The input to this neural network consists of the feature types and their technological attributes such as dimensions, tolerance and surface finish and output in the form of machining operations required for each feature. The second stage deals with further refinement of the manufacturing operations into categories like roughing, semi finishing and finishing operations by another neural network. This neural network takes in as input the feature attributes and the basic machining operations selected in the first stage and it outputs further subdivisions of the operations. The process plan selection is done one feature at a time including within-the-feature process sequencing. The input data necessary for process selection is entered via a user interactive form from a Feature-Based Modelling (FBM) shell.

Devireddy et al (2002) has used a back propagation neural network based approach for selection and sequencing of machining operations for rotationally symmetrical components, containing machining features like hole, step, taper and thread. The process plan selection is done for all the features at a time taking into consideration the global sequencing of machining operations across all the features of the part. The input data for process selection, which consists of the type of the machining feature along with its geometric attributes such as dimensions, tolerance and surface finish, is entered manually and the features are presented in a certain order from the left face of the part.

The neural network based approaches offer several advantages over approaches based on decision trees and expert systems. In situations where the process planning knowledge cannot be easily expressed in explicit rule form, the neural network approach can be used to administer the implicit form of knowledge. It is capable of performing pattern classification tasks by learning arbitrary mappings between the input and desired output patterns from a limited set of examples provided to the network during training. The possible inferences are stored implicitly in the weights of the network. It is characterized by high processing speed once the network is trained, capable of adapting to changing environments through re-training and able to generalize beyond the original set of examples presented in their construction phase and produce meaningful solutions to the problems where input data contains errors or is incomplete.

The approaches based on neural networks, however, have some shortcomings over those of decision trees and expert systems in that they provide no explanation of the rationale behind their inference procedure. Their lack of explicitly stated rules and vagueness in knowledge representation lead to a black box nature. Furthermore, the configuration of the neural network including training is time consuming and the network topology is chosen by a trial and error method.

2.1.4 Hybrid AI Based Approach

Different hybrid AI based approaches have been used for solving the machining process selection problem including a neural-fuzzy technique and a neuro-fuzzy network. The following gives details of each of these hybrid AI based approaches.

Huang et al (1996) has developed a neural-fuzzy technique for hole making machining process selection by incorporating a neural network within a fuzzy reasoning system. Fuzzy rules were developed for hole-making process selection based on process capability information. The factors considered were tolerance, roughness and cost. The above fuzzy rules are used to first convert the roughness and tolerance values of the given hole into fuzzy memberships for cost, corresponding to different hole machining processes, which then constitute the input to the neural network. The neural network is used to finally select the most appropriate hole machining process. The process selected represents the last one needed to produce the desired hole, e.g. if let's say a grinding process is selected, it implies that prior to grinding, drilling and boring process are used to produce the hole surface and enlarge the hole respectively. The neural network is trained using the back propagation algorithm. The authors have found that the neural-fuzzy network is superior compared to a pure neural network in terms of shorter training time and improved prediction accuracy.

Maiyo et al (1999) has developed a neuro-fuzzy network (NFN) based approach to machining operations selection for rotationally symmetric components comprising of shafts, gear wheels and disks. A separate NFN has been used for each feature. The inputs to each NFN are the feature types and their attributes such as surface finish and tolerances. The operation sequences are the outputs. Each NFN has 5 layers. The 1st layer is the input layer. 2nd layer is the fuzzification layer. Layer 3 neurons perform precondition matching of the fuzzy rules and fulfil the AND operation. Layer 4 neurons integrate the fired rules having the consequence by the OR operator. Layer 5 (output layer) performs the defuzzification and weighing of the output that is the machining method and the degree of confidence associated with it. Previous plans are used to generate the fuzzy rules. The supervised learning has been used to train and validate the network structure. The input feature parameters required for process selection need to be entered by the user through dialog boxes.

2.2 Different Approaches for Solving Various Set-up Planning Problems in CAPP

In this section, details of the different generative process planning approaches listed in Table 2.1 for solving various set-up planning problems in CAPP will be briefly discussed along with their advantages and limitations.

2.2.1 Algorithmic and Graph Based Approaches

Different algorithmic and graph based approaches have been used for solving the set-up planning problems. The following gives details of each of these approaches.

Huang et al (1996) used a graph theoretical approach to automatically solve the problem of set-up planning for machining rotational parts. First, all the features present in the part are grouped into three sets according to whether they can be machined from the left, right or both directions. Then the geometric tolerance relations between the different features are represented in a matrix form, from which a weighted tolerance graph is constructed for illustrative purposes. In the graph, each feature is represented as a vertex and the geometric tolerance relation between a pair of features is represented by the edge connecting two vertices in the graph. The graph consists of three sub-graphs to indicate the three sets of features according to their directions of machining. Then the features which can be machined from both the left and the right directions are assigned to one of the sets of features based on their tightest tolerance relationships with other features. This results in two unique sets of features: a set of features, which can be machined from the left and another set of features, which can be machined from the right. These sets are actually the two set-ups and represented by a new tolerance graph with two sub-graphs depicting the two set-ups. After the set-ups have been grouped, the tolerance relationships which could not be satisfied by grouping the features in the same set-up are obtained by proper selection of features to be used as datums for each set-up. For each set-up, a cylindrical and vertical feature must be selected as datums. So the tolerance graph is divided into four sub-graphs (two for each set-up) according to whether they are cylindrical or vertical. Then appropriate datums features are chosen again based on their tightest tolerance relations with features from the other set-up. An algorithm has been developed by the authors using C++ programming language to implement the above approach to set-up planning.

Huang (1998) used an algorithmic approach to automatically solve the problem of set-up planning for machining rotational parts. First of all, the design drawing of the part is examined and the set of features to be machined, the set of features that exist on the stock and the set of features (i.e. the cylindrical surfaces and plane surfaces) that are suitable for locating and clamping considerations are identified. Then the tool approach directions of each feature are identified and represented by a two dimensional vector to indicate if it can be machined from the left or from the right. The geometric tolerance relationships between the different features are identified and represented in the form of a matrix called the adjacency matrix, where each entry in the matrix represents the value of the geometric tolerance between a pair of features; further for illustrative purposes, the geometric tolerance relationships between features are also represented using a weighted undirected graph (which is constructed from the above adjacency matrix) in which each feature is represented as the vertex and the tolerance relationship between two features represented as the edge. The problem of set-up planning is then formulated mathematically as finding the set of features to be machined in each set-up and the set of features to be used for locating and clamping for each set-up, subject to various constraints such as, all features within the same set-up have a common tool approach direction, set-up datums for a certain set-up are not machined in that particular set-up and so on. An algorithm has been described by the author to solve the above problem and generate a feasible set-up plan for machining the part.

Zhang et al (1999) used a hybrid graph approach to solve the problem of set-up planning in CAPP for machining prismatic parts. First the part drawing is examined and the tool approach directions (TAD) for machining the different faces of the part are identified. The tolerance relations between the different faces of the part are represented using two matrices: the adjacency matrix, where each entry in the matrix represents the value of the tolerance relations between a pair of faces and the all-vertex incidence matrix, where each entry in the matrix is used to indicate the type of connectivity between the different vertices of the hybrid graph, which is to be subsequently constructed from the above adjacency matrix and the all-vertex incidence matrix. The hybrid graph is used to graphically represent the tolerance relations between the different faces of the part; each vertex in the graph represents a face and the edge connecting any two vertices represents the tolerance relation between them. Then for those features which have more than one TAD, a unique TAD is assigned to them considering the tolerance relation with other features. Then the faces are grouped into set-ups based on commonality of TADs, tolerance requirements and machining precedence requirements. Then an algorithm, developed by the authors, based on the hybrid graph mentioned above is used for set-up sequencing and datum selection.

To solve the problem of set-up planning for turned parts, Lee et al (2001) used a precedence-directed graph to describe the precedence relations among machining operations followed by a searching strategy to find the various feasible sequences of operations. First the various precedence constraints between operations are identified by the process planner. They include the precedence constraints imposed by universal machining knowledge such as, for example, roughing operations followed by finishing operations and the precedence constraints specified by the process planners during the manufacturability analysis due to technological reasons such as surface accessibility, tolerance specification achievements, etc. Next the machine and the tool used by each operation and the set-up to which each operation belongs, along with their respective costs are identified by the process planner. Then from the above precedence constraints, a precedence-directed graph is constructed. Each operation is depicted by a node in the graph and a path between any two nodes describes the precedence relations among operations. Next a searching strategy developed by the authors is used to automatically search through the graph to find all the feasible operation sequences for the different set-ups. These sequences are further optimised with respect to machine, set-up and tool change costs.

For solving the problem of set-up planning for machined rotational parts, Shunmugam et al (2002) used a Feature Precedence Graph (FPG) followed by a searching strategy that searches through the graph to find the various feasible sequences of machining the features for the two set-ups. First, different production rules based on heuristic and expert knowledge are used to generate various feasibility constraints such as precedence constraints, locating constraints, accessibility constraints, non destruction constraints, geometric tolerance constraints, etc. arising from the inter-relationships that exist between the features. Using these constraints, a Feature Precedence Graph (FPG) is constructed for graphically representing the precedence relationships between features. The features are depicted as nodes of the graph and the two end faces are depicted by the two starting nodes of the graph with null node as their parent. Various information such as the set-up to which a particular feature belongs and the relations between two parent nodes that may be either 'AND' or 'OR' are indicated on the graph. Next a searching strategy developed by the authors is used to automatically search through the graph starting with one of the null nodes to find all the feasible operation sequences for the two set-ups. These sequences are further optimised with respect to feature adjacency, datum/reference features and preference in processing the features in order.

The applications of the above approaches based on algorithms and graphs have been reported to give good and accurate results. However, there are limitations of the above approaches as follows. Firstly, to implement the above approach, the developer has to think of every possibility that may arise. The approach will fail if it encounters cases not covered by the algorithm. Further, they are rather programming-intensive. To be successful, the program must contain all possible combinations of input and output values. In case of an extremely complex problem, the size of a conventional program could become too large and computationally intensive and may need large computing resources. They are inflexible, responding poorly to new situations; for example, if, as a result of changing manufacturing environment, any modification of the current process planning methodology is necessary, it would require rewriting of the original program, which could turn out to be a tedious and time-consuming exercise in certain situations.

2.2.2 Expert Systems Based Approach

Different expert system based approaches have been used for solving the problems of set-up generation and operation sequencing. These include the use of different tools for building expert systems such AI programming languages like LE-LISP and Expert System building tool like Knowledge Engineering Environment (KEE). The following gives details of each of these expert system based approaches.

Wang et al (1988) has used an expert system based approach to solve the problems of determining the operation sequence and set-up formation in set-up planning. In order to determine the operation sequence, four types of constraints have been mainly considered namely, operation precedence constraints, part geometry constraints, tooling constraints and geometric tolerancing constraints. A set of production rules has been developed to describe each of the constraints. The above set of rules has been used to determine the sequence within a group of operations. Further a set of rules have been developed to group the machining operations on surfaces together based on tool types to save the tool change time, and assign the machining operations on surfaces within the same set-up based on geometric tolerance relationships among the surfaces in order to obtain the tolerance requirements specified in the part design.

Joshi et al (1988) has used an expert system based approach to generate the set-ups for machining prismatic parts. Production rules have been developed to establish the precedences between the different machining operations on features based on the datum and reference information given in the part design and the heuristic knowledge on process planning. Also rules have been developed to test if two machining operations have the same tool approach direction, resting face, machines and material condition and assign them to a single set-up and then the above precedence rules are used to determine the sequence of operations within each set-up.

Chang (1990) has used an expert system based approach to solve the different problems in set-up planning for prismatic machined parts. First of all, a set of knowledge based rules is used to generate precedence relationships between features due to various constraints such as process geometric constraint, manufacturing practice, location tolerance and reference (datum) surface constraints. Then the features comprising the part to be machined are grouped into set-ups using tool approach directions as commonality with the objective of minimising the number of set-ups. The precedence between the set-ups is determined using the feature precedences. The features within a set-up are then grouped using cutting tools as the commonality with the objective of minimising the number of tool changes. The operation sequence within the set-up is determined using the feature precedences established earlier. The input data necessary for operation sequencing is obtained from the feature based design model of the part. The overall system was implemented using the expert system shell, KEE.

Sabourin et al (1996) used a combination of expert system and constraint programming based approach for set-up generation for prismatic machined parts containing features such as plain, round, threaded holes, bored holes, face milled flat surfaces and simple pockets. The input for set-up generation is obtained from the feature based design of the part modelled in CATIA. The problem of set-up planning has been defined as one of constraint programming. First all the possible support features and datum features for the set-ups are determined. Then starting with each of the support features, all the possible set-ups are generated one by one. It is accomplished through a set of constraints based on the various possible relationships between features such as technological and topological relationships and kinematic constraints, imposed by the machine kinematics and the type of work holder. The above constraints have been defined by a set of production rules in the system. In order to generate a set-up, the process planner has to first choose a support surface. Then the system submits each machining feature one by one to the above rules which, in turn, activate all the constraints between the feature submitted for consideration and the support feature or datum chosen by the process planner. Then from the constraints generated above, all the features that can be machined in the same set-up with the chosen support surface are determined along with the ordering of the operations needed to machine the features within the set-up. The knowledge base for the expert system was developed with the LE-LISP language.

Kim et al (1998) used a combined expert systems and mathematical programming based approach to solve different problems in set-up planning for machining prismatic parts. A set of rules has been developed to establish the precedences between the operations based on the datum and reference information given in the part design to assign the machining of reference features in the first set-up. Also a set of rules is used to generate the precedence constraints between features based on the heuristic knowledge of process planning and they are used to examine the feasibility of various operation sequences. A mathematical programming model has been developed by the authors to perform the grouping of operations into set-ups and sequencing of operations within the set-up, and a set of rules is used to eliminate the infeasible sequences. Another set of rules is used to assign operations having the same machining directions to the same set-up and clustering operations having tool commonality so as to minimize the number of tool changes.

The major advantages and limitations of expert system based approaches had been discussed in section 2.1.2.

2.2.3 Fuzzy Logic Based Approach

A fuzzy logic based approach was used by Ong et al (1997) for computer-automated set-up planning for machining prismatic parts and castings. It takes as input feature-based descriptions of the part obtained from a feature recognizer. The various feature relations that influence set-up formation and sequencing are established through fuzzy production rules, based on factors such as geometric relationships between features, fixturing and machining requirements, geometric tolerance relationships between features and machining heuristics. These factors are fuzzified through fuzzy membership functions. The fuzzy features relations obtained above together with the set-up planning knowledge, encoded as fuzzy production rules are used to formulate the set-up plan of the part. The overall system was implemented using an expert system shell called KAPPA-PC.

The application of fuzzy logic methods offer a structured and rule based knowledge representation similar to that of expert systems using IF-THEN rules but with linguistic labels. It enjoys a significant advantage over the expert systems in that it is characterized by ability to handle uncertainty and reason with imprecise information.

The main weaknesses of fuzzy logic method, however, are that it is restricted to the fields where expert knowledge is available and the number of input variables is small, and that it is unable to automatically acquire the inference rules like the expert systems, and also the problem of finding appropriate membership functions for the fuzzy variables.

2.2.4 Neural Networks Based Approach

Different neural network based approaches have been used to solve the problems of set-up generation, datum selection and operations and set-up sequencing. These include the use of supervised neural network that uses back-propagation training algorithm for datum selection, unsupervised neural network such as the Kohonen self-organising neural network for clustering features and operations into set-ups, and the Hopfield neural network for operations and set-up sequencing. The following gives details of each of these neural network based approaches.

Chen et al (1993) used a neural network based approach for set-up generation in machining process planning of prismatic parts. An unsupervised neural network was used for clustering the features comprising a part into set-ups based on commonality of cutting tools and tool approach directions of the features so as to minimize the number of tool changes. The input pattern to the neural network consists of all the features of the part along with information such as the tool approach directions of each feature, which is represented by a six-digit ordered binary pattern and the set of tools needed for processing each feature, which is represented by an n-ordered-tuple. After running the neural network, the features are automatically clustered into a number of set-ups in a way as to minimise the number of tool changes.

Mei et al (1995) used a neural network approach for CAPP to automatically select machining datums for rotational parts. It is applicable to machined rotational parts with 10 or fewer surfaces of the types, external cylindrical, vertical and others. In this work, a three layer back propagation neural network with one hidden layer was used. The input to the neural network consists of the shape information of the part, which includes the type of surfaces present coded in a predetermined format and also the tolerance specifications between the different part surfaces. The output of the network gives the surfaces selected for locating and clamping. The input data for datum selection namely the shape information and tolerance specifications have to be entered manually by the user.

Ming et al (2000) have used self-organising neural networks and Hopfield neural networks to solve the set-up planning problem in CAPP. First of all, Kohonen self-organising neural network was used to group the operations into set-ups, taking into consideration constraints such as fixtures/jigs, approach directions, feature precedence relationships and tolerance relationships in the order of importance given above. The input patterns to the Kohonen neural network consist of all the operations that are needed to machine the different features present in the part along with information on each operation such as the type of fixture/jig used, the approach directions and the feature precedence, if any, with other features of the part. After running the neural network, the operations are automatically grouped into a number of set-ups. Once the generation of set-ups is complete, the Hopfield neural network is used to solve the problem of determining the operation sequence within a set-up and the set-up sequence. To solve the problem of operation sequencing within a set-up, constraints such as the feature precedence relationships, natural operation order, position tolerance and cutting tools have been considered. By considering each operation as a city and the constraints among the operations as the distances between the cities, the operation sequencing problem was mapped onto the travelling salesman problem and solved using Hopfield neural network. To solve the set-up sequencing problem, constraints such as the feature precedence relationships, natural operation order, position tolerance and fixture/jigs have been considered. By considering each set-up as a city and the constraints among the set-ups as the distances between the cities, the set-up sequencing problem was mapped onto the travelling salesman problem and solved using Hopfield neural network. After generation of the operation sequences and setup sequences, they are checked manually for any violation of the position tolerance relationships between features and the sequences are altered slightly, if necessary, to arrive at the final operation sequence and set-up sequence.

Chang et al (2001) used a Hopfield neural networks based approach for operation sequencing in set-up planning for prismatic parts. For a given part, first all the possible part orientations are identified and feature accessibilities of each part orientation are determined. Then the part orientations that can cover all the manufacturing features with a minimal number of changes in part orientations are identified and the features are accordingly assigned to these part orientations such that they meet the requirements of precedence relationships, tool commonality and approach directions. Then the different table orientations that are possible for each of the above part orientations are identified, the feature accessibility for each table orientation in its corresponding part orientation is determined and the table orientations that best cover all the manufacturing features with a minimal number of changes in table orientations are obtained. The manufacturing features are then accordingly assigned to each table orientation such that they meet the requirements of precedence relationships, tool commonality and approach directions. The tool travel distances between features of the above part orientations are calculated and expressed in the form of a matrix. Next the preference position matrix is determined for each part orientation. It gives the preferable positions of each feature in the machining sequence for each part orientation, such as for example, if there are datum features present, then obviously they have to machined first. Then by considering each feature within a set-up as a city and the tool travel distance between the features as the distance between the cities, the problem of sequencing the features is mapped onto the traveling salesman problem and solved using the Hopfield neural network. The output from the Hopfield neural network is the manufacturing sequence with the minimum tool travel distance between features.

The principal advantages and limitations of neural network based approaches had been discussed in section 2.1.3.

2.2.5 Hybrid AI Based Approach

A hybrid AI approach based on Hopfield neural net coupled with simulated annealing has been used by Chen et al (1998) to solve the problem of set-up planning for prismatic parts. By considering each feature as a city and the set-up time due to set-up and tool changes as the distance between the cities and the precedence and the critical tolerance relationships between features as the constraints, the feature sequencing problem was mapped onto a constraint based travelling salesman problem (TSP) and solved using the Hopfield neural network approach. A Hopfield energy function has been formulated to solve the problem with the constraints considered as additional penalty functions. An algorithm based on simulated annealing was adopted to search for the minimum energy state of the net while avoiding the local minima. The feature sequence obtained aims at minimising the number of set-ups and tool changes while ensuring little or no violation of feature precedence relationship and, thus keeping the critical tolerance violation to a minimum. After determining the feature sequence, the set-ups are generated from the sequenced features using a vector intersection approach based on common tool approach directions.

2.3 Comments on the Literature Review

The above literature review on various approaches for automating the machining operations selection shows that the neural networks based approaches can overcome several limitations of the decision trees and the expert system based approaches. They are capable of automatically acquiring knowledge in the form of example data, do not require the domain knowledge to be presented in explicit rule form, and they are able to generalize beyond the original set of training examples and produce meaningful solutions to the problems where input data contain errors or are incomplete. Modification of knowledge base can be accomplished easily through retraining of the network. Owing to their parallel architecture, they are characterized by high processing speeds, once they are trained, which leads to faster inference compared to the decision trees and expert systems. However, inspite of the above advantages, the neural network based approaches adopted by Knapp et al (1992) and Devireddy et al (1999, 2002) could have a set of problems of their own that includes choosing the set of training examples that is best representative of the problem domain. It is difficult and a tedious and time-consuming exercise of trial and error. There are no guidelines at present on how to choose them and the values of input variables are often randomly selected from the entire range of values possible. Sometimes even after a large set of examples has been chosen, there may remain some cases where small sets of exceptions may be either unrepresented or poorly represented as a result of which such cases may be very difficult to handle correctly. Also an issue that has not been adequately addressed by previous researchers is that whether any prior domain knowledge on machining operations selection could be taken advantage of. The prior domain knowledge is already well known to reduce the complexity of learning in ANN. Further most of the previously developed neural network models tend to recommend a single machining operation sequence for a given feature of the part. But, in practice, the process planning system needs to know the alternative ways of machining a part for use in developing process planning alternatives. Keeping the above in mind, the author has developed a back-propagation neural network methodology for selection of all possible alternative operation sequences for machining a given feature in rotationally symmetrical parts. The neural network has been pre-structured with prior knowledge on machining operations selection in the form of heuristic or thumb rules. During the preparation of training examples, the above thumb rules have been used to serve as guidelines with the input patterns of the training examples chosen in such a way that they activate one or more of these thumb rules. There are some benefits that can be gained by prestructuring the input data representation format of the neural network by prior knowledge in this way. As will be seen later, it simplifies the process of preparation of training examples and helps to better ensure that the entire problem domain is represented.

The literature review on approaches for automating the set-up planning indicates that the expert system has lot of potential for solving the different problems of set-up planning. But its potential for application for set-up planning in rotational parts has yet to be fully explored. Most of the previous research efforts in this regard by Wang et al (1988), Joshi et al (1988), Chang (1990), Sabourin et al (1996), Kim et al (1998) and others have been limited to the prismatic parts domain. For set-up planning in rotational parts, however, a different approach needs be adopted. This is so because in case of machining rotational parts, each feature can have only 2 possible TADs, i.e. the left and/or the right while for prismatic parts, each feature may have upto 6 possible TADs. As a result, only two set-ups are possible for machining rotational parts, whereas there are more than two set-ups possible for prismatic parts. Also, in most of the previously reported expert systems based approaches, a mixture of the expert systems and some algorithmic approach had been adopted to solve set-up planning problems. The limitation of it is that the resulting system tends to be inflexible, responding poorly to new situations. In particular, when it comes to modification of the current set-up planning knowledge or acquiring new knowledge, it might require rewriting of the original program for the algorithm. Keeping the above in mind, the author has developed an expert system based methodology for set-up planning in CAPP systems for machined rotationally symmetrical parts to solve the three main set-up planning problems, namely set-up formation, operations sequencing and selection of locating and clamping surfaces for each set-up.

CHAPTER 3

PROPOSED NEURAL NETWORK BASED METHODOLOGY FOR SELECTION OF MACHINING OPERATION SEQUENCES

The neural network methodology developed by the author for selection of machining operations sequences has been described in detail in the present chapter. It caters to the process planning of machined rotationally symmetrical parts. The commercial PC-based software package *NeuFrame Version* 4 (2000) has been used to simulate the neural network operation It takes in, as input, the attributes of each feature and is capable of automatically selecting all the possible alternative sequences of machining operations to produce that feature. The following key issues regarding development of the neural network based methodology have been discussed:

a) gathering of domain knowledge for formulating the thumb rules,
b) topology of the network,
c) format of representation of the input and output decision variables,
d) training and validation of the neural network, and
e) how to accomplish modification of the knowledge base so as to adapt it to new situations.

The potential for application of the developed neural network methodology has been illustrated with the help of some examples of rotationally symmetrical parts and the results and discussions have been presented.

3.1 Description of the Developed Neural Network Methodology

The following discusses the key issues mentioned above regarding development of the neural network based methodology.

3.1.1 Gathering of Domain Knowledge for Formulating the Thumb Rules

A set of thumb rules has been developed for selection of machining operation sequences for different types of features commonly encountered in rotationally symmetrical parts. The necessary domain knowledge for formulating the rules was collated from various sources such as handbooks and textbooks on machining [Bralla (1986), Halevi et al (1995) and Wang et al (1991)]. But in practice for a given company, it may be necessary to consult the appropriate vendor catalogues of the manufacturing equipments present in the shop floor and company specific manufacturing process handbooks for detailed information about process capabilities of various machining operations. These catalogues and handbooks provide, among other information, the dimensions, the dimensional tolerances and the surface finish ranges attainable by different machining processes. Much of this information is in the tabular form or in the form of charts. Typical use of a handbook involves, first of all, given a feature to be machined, locating the pertinent section and table or chart in the handbook to obtain the tolerance and surface finish requirements specified on the feature and then select the appropriate machining operation or a sequence of operations as necessary. The thumb rules that have been developed are expressed in the following form:

IF (Feature is of the type Feat) AND (Dimension of the Feature is Dim_i) AND … (Tolerance of the Feature is Tol_j) AND … (Surface finish of the Feature is SF_k), THEN (Operation sequence is $OpSeq_l$)

In the antecedent 'IF' part of the rule, Feat represents the feature type such as hole, external diameter and so on; the subscript i of a Dim_i represents a certain range of dimenions of the feature such as for example, 0-3 mm; the subscript j of a Tol_j represents a certain range of tolerances such as for example, 3-10 μm; the subscript k of a SF_k represents a certain range of surface finish such as for example, 0.04-1.25 μm. In the consequent 'THEN' part of the rule, subscript l of an $OpSeq_l$ represents a feasible operation sequence for machining a certain feature such as for example, drilling-boring-honing for machining a hole feature.

The types of the features that have been considered are internal surfaces of revolution like holes, external surfaces of revolution like external diameters, external taper, external thread, groove, face, and other features such as slot, keyway, which are commonly encountered features in rotationally symmetrical parts. The ranges of dimensions, tolerances and surface finish considered for the above features are summarised in Table 3.1. Table 3.2 lists the 33 operations sequences that have been considered for machining the above features. An extract from the thumb rules is shown in Figure 3.1. These rules have to be learnt by the neural network model.

Table 3.1. Ranges of Dimension, Tolerance and Surface Finish Considered for Different Features

Feature type	Dimensions (Diameter or Width)	Tolerance	Surface finish Ra
Hole	Up to 50mm (Length/Diameter ratio upto 10)	3-390μm	0.04-80μm
External step	Up to 50mm	4-390μm	0.08-80μm
Groove	Up to 50mm	40-250μm	2.5-20μm
Face	Up to 50mm	10-390μm	1.25-80μm
Slot	Up to 6mm	6-190μm	0.32-20μm
External taper	Up to 50mm	4-390μm	0.08-80μm
External thread	Up to 50mm	10-390μm	1.25-80μm

Table 3.2. Operation Sequences Considered for Machining the Different Features

Operation Sequence	Used for machining	Operation Sequence	Used for machining
Drill	Hole	Groove turning (one pass)	Groove
Drill-Counter Bore		Groove turning (two passes)	
Drill-Counter Bore-Rough Ream-Semi finish Ream		Rough turn	Face
Drill-Rough Bore		Rough turn-Semi finish turn	
Drill-Rough Bore-Semi finish bore		Rough turn-Semi finish turn-Finish turn	
Drill-Rough Bore-Semi finish Bore-Finish Bore		Rough mill	Slot
Drill-Rough Bore-Semi finish Bore-Rough Grind-Semi finish Grind		Rough mill-Semi finish mill	
Drill-Rough Bore-Semi finish Bore-Rough Grind-Finish Grind		Rough mill-Semi finish mill-Finish mill	
Drill-Rough Bore-Semi finish Bore-Grind-Hone		Rough Turn	External taper
Deep hole drill		Rough Turn-Semi finish turn	
Rough Turn	External step	Rough Turn-Semi finish Turn-Finish Turn	
Rough Turn-Semi finish turn		Rough Turn-Semi finish Turn-Rough Grind	
Rough Turn-Semi finish Turn-Finish Turn		Rough Turn-Semi finish Turn-Rough Grind-Semi finish grind	
Rough Turn-Semi finish Turn-Rough Grind		Rough Turn-Semi finish Turn-Rough Grind-Finish Grind	
Rough Turn-Semi finish Turn-Rough Grind-Semi finish Grind		Rough Turn-Threading	External thread
Rough Turn-Semi finish Turn-Rough Grind-Finish Grind		Rough Turn-Semi finish turn-Threading	
		Rough Turn-Semi finish Turn-Finish Turn-Threading	

IF (Feature is a Hole) AND (Diameter of the Hole is 0-3 mm) AND (Tolerance of the Hole is 3-10 μm) AND (Surface finish of the Hole is 0.04-1.25 μm), THEN (Operation sequence is Drilling-Rough Boring-Semi finish Boring-Grinding-Honing).
IF (Feature is a Hole) AND (Diameter of the Hole is 0-3 mm) AND (Tolerance of the Hole is 4-6 μm) AND (Surface finish of the Hole is 0.08-0.16 μm), THEN (Operation sequence is Drilling-Rough Boring-Semi finish Boring-Rough Grinding-Finish Grinding).
IF (Feature is a Hole) AND (Diameter of the Hole is 0-3 mm) AND (Tolerance of the Hole is 6-10 μm) AND (Surface finish of the Hole is 0.16-0.63 μm), THEN (Operation sequence is Drilling-Rough Boring-Semi finish Boring-Rough Grinding-Semi Finish Grinding).

Figure 3.1. Extract from the Set of the Thumb Rules on Selection of Machining Operations Sequences for Holes

3.1.2 Topology of the Neural Network Model

The topology of the proposed neural network model is shown in the Figure 3.2. For the problem at hand, a feed forward architecture of the neural network has been chosen. It comprises of an input layer of neuronal nodes to represent the input decision variables, an output layer of neuronal nodes to represent output decision variables and one or more hidden layer of neuronal nodes. The detailed explanation on the format of representation of the input and output decision variables for the neural network will be given in the following section.

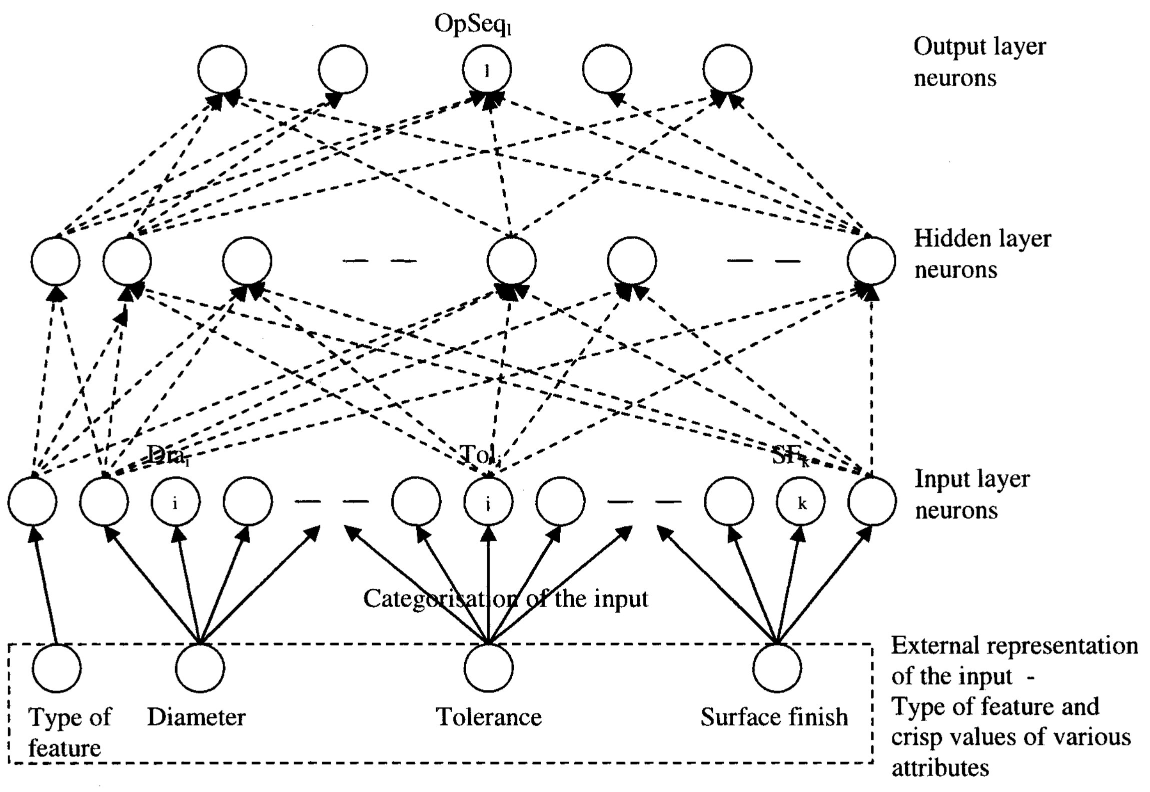


Figure 3.2. Topology of the Proposed Neural Network Model

3.1.3 Format of Representation of the Input and Output Decision Variables for the Neural Network

The input decision variables for the problem at hand consist of the type of feature (such as hole, external step, etc.) and various attributes namely, diameter or width, tolerance and surface finish of the feature under consideration. The type of feature has been represented by integer values from 1 to 7 to denote the seven features under consideration namely, holes, external steps, groove, face, slot, external taper and external thread respectively. The feature attributes have been represented by their respective numerical values, such as diameter or width of the feature expressed in mm, tolerance expressed in μm and surface finish Ra, expressed in μm. The crisp values of these four variables together constitute the external representation of the input to the neural network as shown in the Figure 3.2. For example, the external representation of input for a hole feature of diameter 5 mm, tolerance 5 μm and surface finish 0.04 μm is given by the following input vector.

Column number	1	2	3	4
Value	1	5	5	0.04

In the above, column number 1 stands for the type of feature (hole in this case), 2 stands for the dimension (diameter in this case), 3 stands for the tolerance and 4 stands for the surface finish of the feature. Next this external representation has to be translated into the format of internal representation of input before presenting it to the input layer of the neural network. In other words, the crisp values of various feature attributes shown above have to be categorized into sets of diameter, tolerance and surface finish (as shown in the Figure 3.3) that correspond to all the possible different ranges of dimension, tolerance and surface finish, encountered in the antecedent 'IF' part of the thumb rules for selection of machining operation sequences. This is accomplished with the aid of simple classification rules as will be explained below. In Figure 3.3(i), 1, 2, 3 and so on stand for the sets of diameter of the hole corresponding to various ranges such as, 0-3 mm, 3-6 mm, 6-10 mm and so on respectively, found in the antecedent 'IF' part of the rules. In Figure 3.3(ii), 1, 2, 3, 4 and so on stand for the sets of tolerance corresponding to various ranges such as, 3-10 μm, 4-6 um, 6-10 μm, 10-14 μm and so on respectively, found in the antecedent 'IF' part of the rules. In Figure 3.3(iii), 1, 2, 3 and so on stand for the sets of surface finish corresponding to various ranges such as, .04-1.25 μm, .08-.16 μm, .16-.63 μm and so on respectively, found in the antecedent 'IF' part of the rules. It may be noted that in some cases, the ranges represented by the sets are overlapping.

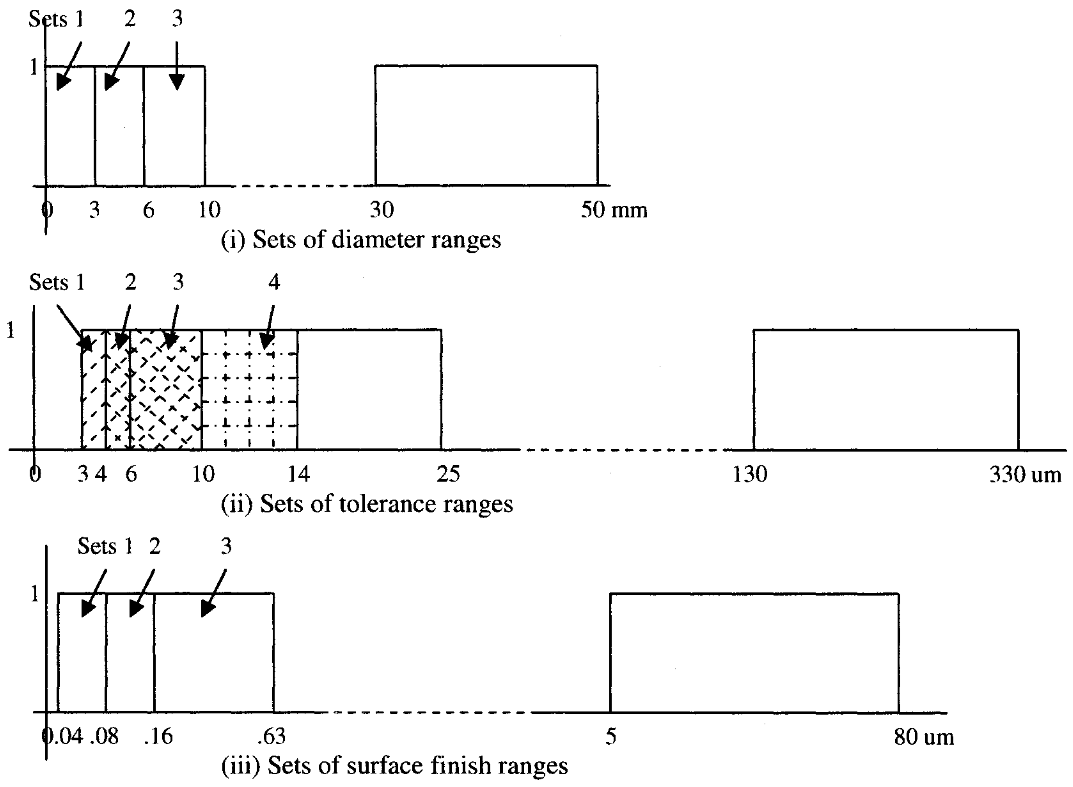

Sets 1 2 3
1
0 3 6 10 30 50 mm
(i) Sets of diameter ranges
Sets 1 2 3 4
1
0 3 4 6 10 14 25 130 330 um
(ii) Sets of tolerance ranges
Sets 1 2 3
1
0 .04 .08 .16 .63 5 80 um
(iii) Sets of surface finish ranges

Figure 3.3. Sets to Represent the Diameter, Tolerance and Surface Finish Ranges Encountered in the Thumb Rules for Operations Sequence Selection

Example of some rules that have been used for classification of the crisp values of diameter, tolerance and surface finish into their corresponding sets are given below. For example, suppose that the diameter ranges encountered in the antecedent 'IF' part of the rules are 0 to 3 mm, 3 to 6 mm and so on, then the rules like the ones shown below may be used for assigning diameter values to the corresponding diameter sets.

IF (feature is a hole) AND (its diameter lies between 0 and 3 mm), THEN (it is assigned to the diameter set for hole, 0-3 mm with a membership value of 1 or otherwise 0).
IF (feature is a hole) AND (its diameter lies between 3 and 6 mm), Then (it is assigned to the diameter set for hole, 3-6 mm with a membership value of 1 or otherwise 0) and so on.

In a similar manner, rules like the ones shown below may be used for assigning tolerance values to the corresponding tolerance sets.

IF (feature is a hole), (its diameter lies between 0 and 3 mm), AND (its tolerance lies between 3 and 10 μm), Then (it is assigned to the tolerance set for hole, 3-10 μm in the diameter range between 0 and 3 mm, with a membership value of 1 or otherwise 0).
IF (feature is a hole), (its diameter lies between 0 and 3 mm), AND (its tolerance lies between 4 and 6 μm), Then (it is assigned to the tolerance set for hole, 4-6 μm in the diameter range between 0 and 3 mm, with a membership value of 1 or otherwise 0) and so on.

Similarly, rules like the ones shown below may be used for assigning surface finish values to the corresponding surface finish sets.

IF (feature is a hole) AND (its surface finish lies between 0.04 um and 1.25 μm), Then (it is assigned to the surface finish set for hole, 0.04-1.25 μm with a membership value of 1 or otherwise 0).
IF (feature is a hole) AND (its surface finish lies between 0.08 um and 0.16 μm), Then (it is assigned to the surface finish set for hole, 0.08-0.16 μm with a membership value of 1 or otherwise 0) and so on.

For example, if a hole of diameter 5 mm, tolerance 5 μm and surface finish 0.04 μm is encountered in the external representation of the input, then type of the feature will be assigned the value of 1 (to denote that the feature is a hole), the diameter value will be assigned to the diameter set for holes, 3-6 mm with a membership value of 1, while its membership value for each of the other diameter sets, such as 0-3 mm and for all other sets for which it is outside their ranges will be 0; its tolerance value will be assigned to the tolerance set for holes, namely 4-12 μm and 5-8 μm, each with a membership value of 1 while its membership value for all other sets for which it is outside their ranges will be 0; its surface finish value will be assigned to the surface finish set for holes, namely 0.04-1.25 μm with a membership value of 1, while its membership value for the set, 0.08-0.16 μm and for all other sets for which it is outside their ranges will be 0. Thus as each input is entered, the rules as explained above automatically assigns it to the corresponding sets with a membership value of either 1 or 0 depending on whether it belongs to the set or not.

The input layer of neurons in the neural network is then designed in such a way that one neuronal node is allocated for the type of feature and one neuronal node is allocated to each of the above sets of feature attributes namely diameter or width, tolerance and surface finish. Also the values of all the input layer neurons are normalised to lie between 0 and 1. The number of neuronal nodes in the input layer is equal one (to represent the type of the feature) plus the number of all the possible different ranges of feature attributes encountered in the antecedent part of the rules. In the thumb rules developed, 38 different ranges of diameter, 168 different ranges of tolerance and 33 different ranges of surface finish are found in the antecedent 'IF' part of the rules. So the number of neuronal nodes in the input layer is 240 (i.e. 1+38+168+33). With these 240 neuronal node values, the machining feature and its attributes are represented as a vector of 240 elements and this forms the input pattern to the neural network. The input pattern, given a hole feature of diameter value 5 mm, tolerance 5 μm and surface finish 0.04 μm, when represented in the above format takes the form of the following vector:

Column number	1	2	3	4	5	6	7	8	9	10	..	41	42	..	..	209	..	240
Value	1	0	1	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0

In the above vector, the column number 1 stands for the type of feature, column numbers [2-7], [8-14], [15-19], [20-25], [26-27], [28-34], [35-39] stand for the sets corresponding to the different ranges of diameter of the hole, external step, groove, face, slot, external taper and external thread respectively. Column numbers [40-93], [94-123], [124-135], [136-153], [154-159], [160-189], [190-207] stand for the sets corresponding to the different ranges of tolerance of the above seven features respectively. Column numbers [208-217], [218-223], [224-225], [226-228], [229-231], [232-237], [238-240] stand for the sets corresponding to the different ranges of surface finish of the above seven features respectively.

The output decision variables for the problem at hand comprise of the various feasible machining operation sequences by which the features, which are considered here, can be produced. Within-the-feature operation sequencing is built into each operation sequence. The output layer of the neural network is designed in such a way that one neuronal node is allocated to each of the various feasible machining operation sequences found in the consequent 'THEN' part of the thumb rules. Each neuronal node in the output layer either assumes a nonzero value (less than or equal to 1) to indicate suitability of a certain operation sequence or a value of zero otherwise. The number of neuronal nodes in the output layer is equal to the number of all the feasible machining operation sequences that are found the consequent 'THEN' part of the rules. In the thumb rules developed, 33 different operation sequences were found in the consequent part of the rules. So the number of neuronal nodes in the output layer is 33. With these 33 neuron values, the feasible alternative machining operation sequences are represented as a vector and this forms the output pattern to the neural network. For machining the hole of diameter 5 mm, tolerance 5 μm and surface finish 0.04 μm, the operation sequence is Drilling - Rough Boring - Semi finish Boring – Grinding - Honing, which is represented in the above format by the following vector:

Column number	1	2	3	4	5	6	7	8	9	..	..	33
Value	0	0	0	0	0	0	0	0	1	0	0	0

In the above vector, each of the column numbers [1-10], [11-16], [17-18], [19-21], [22-24], [25-30] and [31-33] stand for a feasible operations sequence for machining the different features namely hole, external step, groove, face, slot, external taper and external thread respectively.

3.1.4 Training and Validation of the Neural Network

The following describes the method of training and validation of the neural network.

3.1.4.1 Training of the Neural Network

The standard back propagation algorithm is used as the learning mechanism for the neural network. Since it is a supervised learning algorithm, both the input and output patterns have to be included in the training examples. The input pattern consists of the values of the input decision variables for which the value of the output decision variables are known and the output pattern or target pattern consists of the target or desired values of the output decision variables. The input decision variables, for the problem at hand, are the type of feature and its attributes, namely diameter or width of the feature and its tolerance and surface finish, and the output decision variables are the various feasible machining operation sequences to produce the features under consideration. The input patterns of the training examples have to be chosen in such a way that they cover the entire range of values of the input decision variables possible in the problem domain. The method of preparation of training examples has been illustrated in the following section.

3.1.4.2 Preparation of Training Examples

The training examples have been prepared using the same set of thumb rules used earlier to prestructure the input data representation format of the neural network.

Table 3.3 shows a training examples dataset that was prepared using the thumb rules shown in Figure 3.1.

Table 3.3. Examples of Input and Output Patterns for the Machining Operations Selection Problem

No.	Input pattern				Output pattern								
	1	2	3	4	1	..	6	7	8	9	10	..	33
1	1	2	3	0.04	0	0	0	0	0	1	0	0	0
2	1	2	3	0.063	0	0	0	0	0	1	0	0	0
3	1	2	3	0.08	0	0	0	0	0	1	0	0	0
4	1	2	3	0.16	0	0	0	0	0	1	0	0	0
5	1	2	3	0.63	0	0	0	0	0	1	0	0	0
6	1	2	4	0.04	0	0	0	0	0	1	0	0	0
7	1	2	4	0.063	0	0	0	0	0	1	0	0	0
8	1	2	4	0.08	0	0	0	0	1	1	0	0	0
9	1	2	4	0.16	0	0	0	0	0	1	0	0	0
10	1	2	4	0.63	0	0	0	0	0	1	0	0	0
11	1	2	6	0.04	0	0	0	0	0	1	0	0	0
12	1	2	6	0.063	0	0	0	0	0	1	0	0	0
13	1	2	6	0.08	0	0	0	0	0	1	0	0	0
14	1	2	6	0.16	0	0	0	1	0	1	0	0	0
15	1	2	6	0.63	0	0	0	0	0	1	1	0	0

The input pattern of each training example, in its external representation format, has 4 columns: first column to indicate the type of feature (the value of 1 denotes that the feature under consideration is a hole) and the remaining three columns to indicate the various feature attributes namely, diameter or width, tolerance and surface finish respectively. The output pattern of each training example has 33 columns, each of which stands for a feasible machining operation sequence to produce a certain feature as already explained in section 3.1.3. The following will explain how the above training examples have been arrived at.

From the antecedent 'IF' part of the rules given in Figure 3.1, it can be found that the feature under consideration is a hole, the diameter range under consideration is 0-3 mm, the tolerance ranges under consideration are 3-10 μm, 4-6 μm and 6-10 μm and the surface finish ranges under consideration are 0.04-1.25 μm, 0.08-0.16 μm and 0.16-0.63 μm. The input patterns for the training examples have been chosen in such a way that they cover the entire range of the feature type, diameter, tolerance and surface finish found in the antecedent part of the above rules. From Table 3.3, it can be found that for all the training examples, a dimension of 2 mm has been chosen as the hole diameter (see Column 2 of the Input pattern). It may be recalled that there exists a neuronal node in the neural network for each of the above ranges of diameter, tolerance and surface finish. So as the value of hole diameter is entered as 2 mm, it is automatically assigned to the neuronal node for the set corresponding to the diameter range 0-3 mm with a membership value of 1 using the classification rule discussed in section 3.1.3; clearly it is sufficient to represent all the possibilities in the diameter range of 0-3 mm. It may be noted that in place of 2 mm, we could have chosen any value between 0 and 2.99 mm as a representative value of the hole diameter for the above training examples. Also for the above training examples three representative values have been chosen for tolerance of the hole namely, 3 μm, 4 μm and 6 μm in order to represent the three sets corresponding to the three tolerance ranges namely, 3-10 μm, 4-6 μm and 6-10 μm respectively. On entering the values of tolerance of the hole as above, the classification rules assign the tolerance value of 3 μm to the neuronal node for the set representing the tolerance range, 3-10 μm with a membership value of 1, the tolerance value of 4 μm to the neuronal nodes for the sets representing the tolerance ranges of 3-10 μm and 4-6 μm, each with a membership value of 1 and the tolerance value of 6 um to the neuronal nodes for the sets representing the tolerance ranges of 3-10 μm and 6-10 μm each with a membership value of 1. Clearly these three values of tolerance together are sufficient to represent the entire range of possibilities of tolerance found in the rules above. Also in the above training examples, three representative values have been chosen for the surface finish of the hole namely, 0.04 μm, 0.08 μm and 0.16 μm in order to represent the three sets corresponding to the surface finish ranges namely, 0.04-1.25 μm, 0.08-0.16 μm and 0.16-0.63 μm that are sufficient to cover the entire range of possibilities of surface finish found in the above rules. Then by different combinations of these values of feature type, diameter, tolerance and surface finish, the set of training examples given in Table 3.3 has been arrived at.

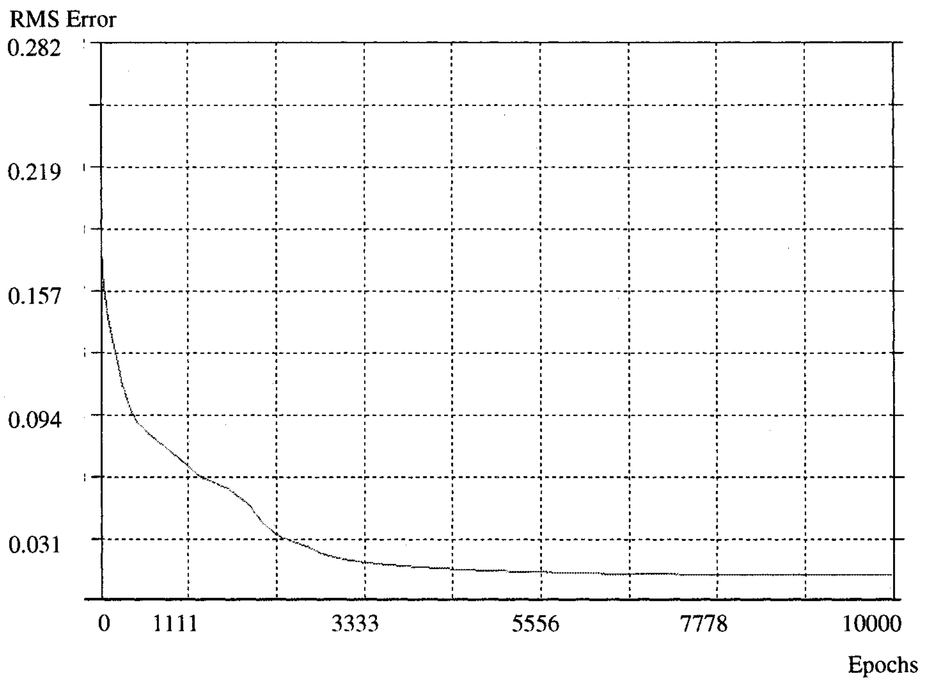
For the problem at hand which is to select the machining operation sequences for different features, a set of 318 training examples were developed using the set of thumb rules developed in section 3.1.1 in the manner explained above. While preparing the training examples, care was taken to make sure that the input patterns of the training examples are chosen in such a way that they cover the entire range of values of each of the four input decision variables found in the rules. The following section describes how the neural network was trained.

3.1.4.3 Method of Training of the Neural Network

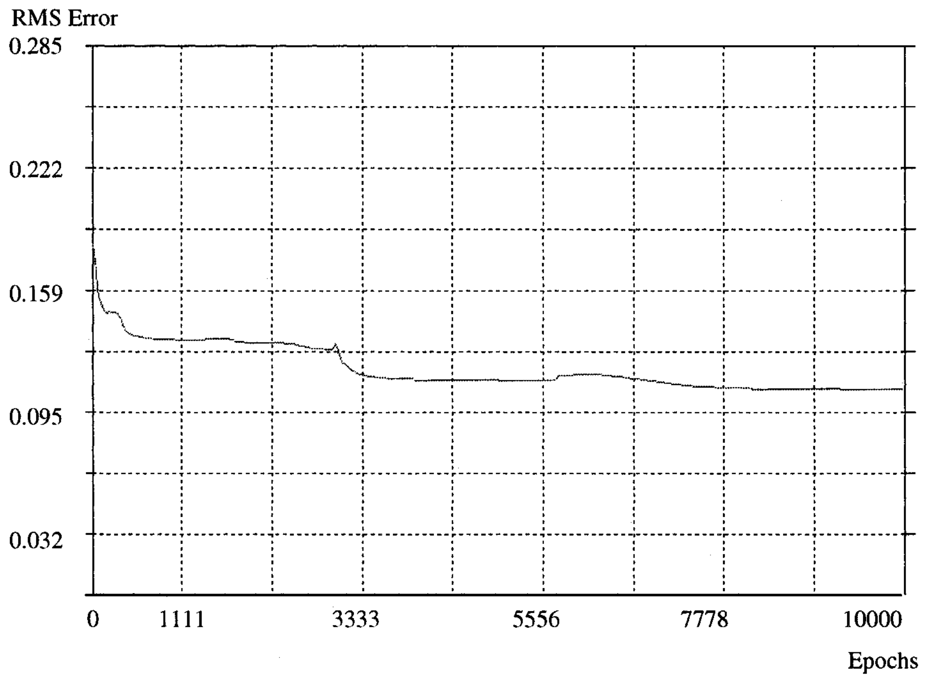
The neural network was trained using the standard back-propagation algorithm. The commercial PC-based software package *NeuFrame Version* 4 (2000) has been used to simulate the neural network operation. Details on creating the back-propagation neural network using NeuFrame are given in Appendix A. There are two alternative training modes possible depending on the particular way of presenting the training patterns to the neural network and depending on when the network weights are updated. They are the pattern mode and the epoch mode. In the pattern mode, the network weights are updated after presentation of each training pattern to the neural network. The root mean squared (RMS) error of the pattern is calculated for each iteration as PE =

$\sqrt{\{\Sigma(T_i - Out_i)^2\}}$, where T_i and Out_i are the target output and the computed output at the neuronal node i in the output layer. The training examples are repeatedly presented until a predetermined level of error E (according to some error criterion) is achieved, in other words, $PE \leq E$ when the training is stopped. In the epoch mode, all the patterns are presented to the network; such a training session over the entire set of examples is called an epoch; after one complete epoch, the network error terms for each pattern is summed and the weights are updated based on an average error across all the training patterns. Training is stopped only when the average error over all the patterns EE falls below E given by $\Sigma PE/n = EE \leq E$, where n stands for the number of all training patterns.

The general strategy adopted for finding out which of the two training modes performs better for the problem at hand is as follows. The neural network is trained using both the pattern mode and the epoch mode in order to characterize the performance differences between the two modes. For the input layer, a linear transfer function was used, whereas for the hidden layer and the output layer sigmoidal transfer functions were used. A total of 8 neuronal nodes were used in the hidden layer. The momentum rate and learning rates used for training the neural network were 0.8 and 0.2 respectively. The training error graphs for the two training modes after 10000 iterations are shown in Figure 3.4. The results of the two training modes are shown in Table 3.4. It shows that the pattern mode of training leads to faster convergence for the problem at hand. So the pattern mode was selected for training the neural network.



(a) Pattern Training Mode



(b) Epoch Training Mode

Figure 3.4. Training Error Graphs for Pattern Training Mode and Epoch Training Mode

Table 3.4. Network Performance for Different Training Modes (No. of Hidden Layer Nodes=8, Momentum Rate=0.8 and Learning Rate=0.2)

Training Mode	RMS Error after 10000 iterations
Pattern mode	0.0133
Epoch mode	0.1070

The general strategy adopted for finding the optimal parameters of the neural network that control the training process is as follows. For each trial, number of hidden layer nodes, random initial weights and biases of the neural network are varied. The neural network is trained using the pattern mode with different combinations of momentum terms and learning rates in an attempt to identify the neural network that performs best on the testing data. The momentum terms used are 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8 and 0.9, whereas the learning rates used are 0.005, 0.1, 0.2, 0.4, 0.5, 0.6, 0.7, 0.8 and 0.9. Since the back-propagation training algorithm uses a first-order gradient descent technique to adjust the connection weights, it may get trapped in a local minimum if the initial starting point in weight space is unfavourable. Consequently, the neural network that has optimum momentum term and learning rate is retrained a number of times with different initial weights and biases until no further improvement occurs.

The impact of the number of hidden layer nodes on neural network training is shown in Figure 3.5. It shows that the network with 9 hidden layers nodes has the lowest error. For networks with number of hidden layer nodes greater than 9, no significant improvement of error occurred. Therefore, the network with 9 hidden layer nodes was considered optimal for the problem at hand.

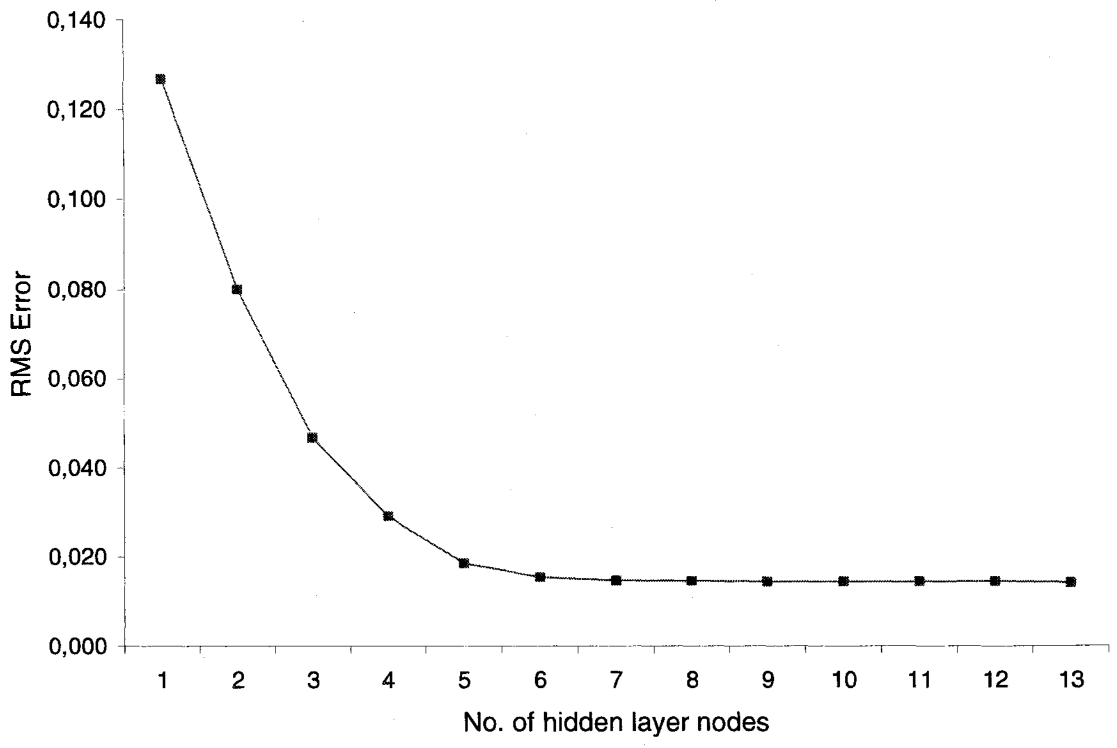


Figure 3.5. Effect of Number of Hidden Layer Nodes on Neural Network Training (No. of Iterations=6000, Momentum Rate=0.8 and Learning Rate=0.2)

The effect of the parameters controlling the back propagation algorithm i.e., momentum term and learning rate on neural network training is shown in Figures 3.6 and 3.7, respectively.

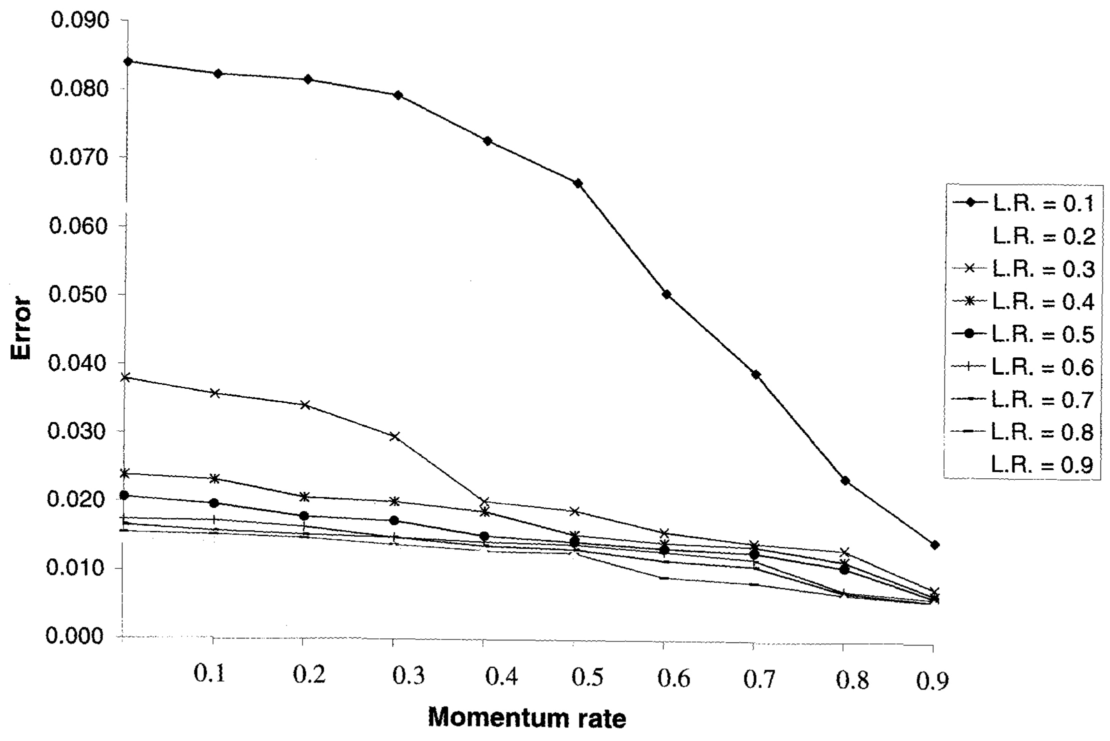


Figure 3.6. Effect of Momentum Rates on Neural Network Training (No. of Hidden Layer Nodes=9, No. of Iterations=6000) Note: L.R. Stands for the Learning Rate

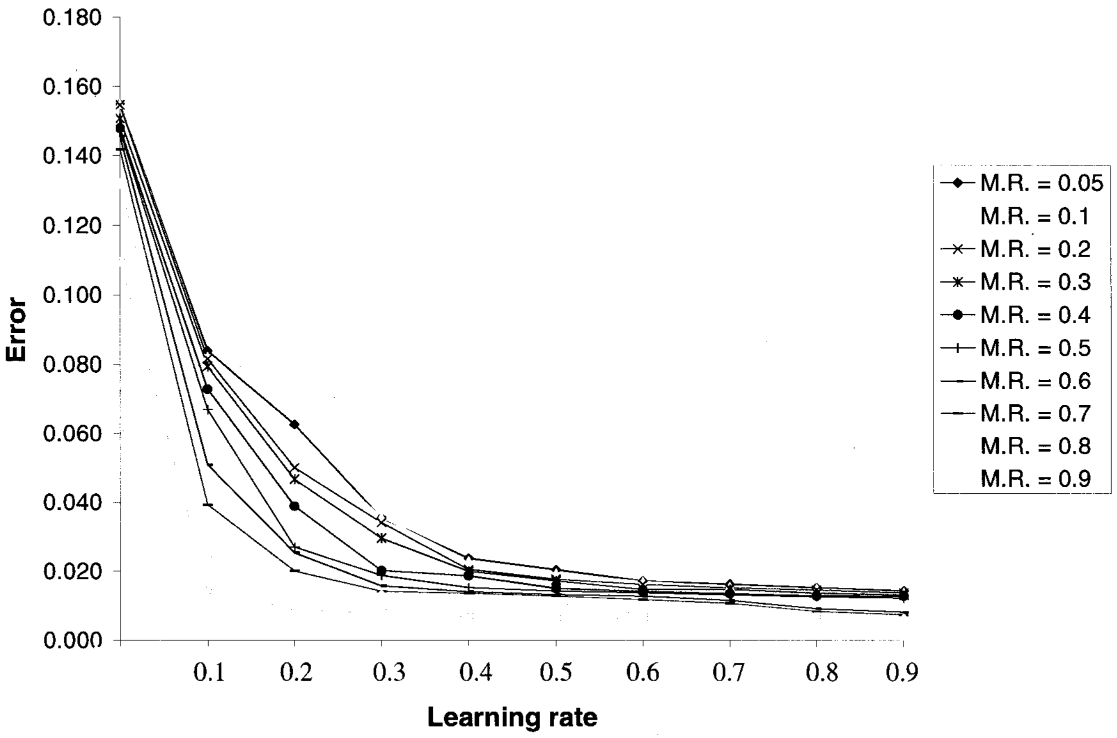


Figure 3.7. Effect of Learning Rates on Neural Network Training (No. of Hidden Layer Nodes=9, No. of Iterations=6000) Note: M.R. Stands for the Momentum Rate

After a number of trials, the following optimum architecture of the neural network, mode of training and the optimum network internal parameters that control the training process were arrived at.

Number of hidden layers	1
Number of nodes in the hidden layer	9
Mode of training	Pattern
Learning rate	0.4
Momentum rate	0.9

The optimum architecture of the neural network is shown in Figure 3.8.

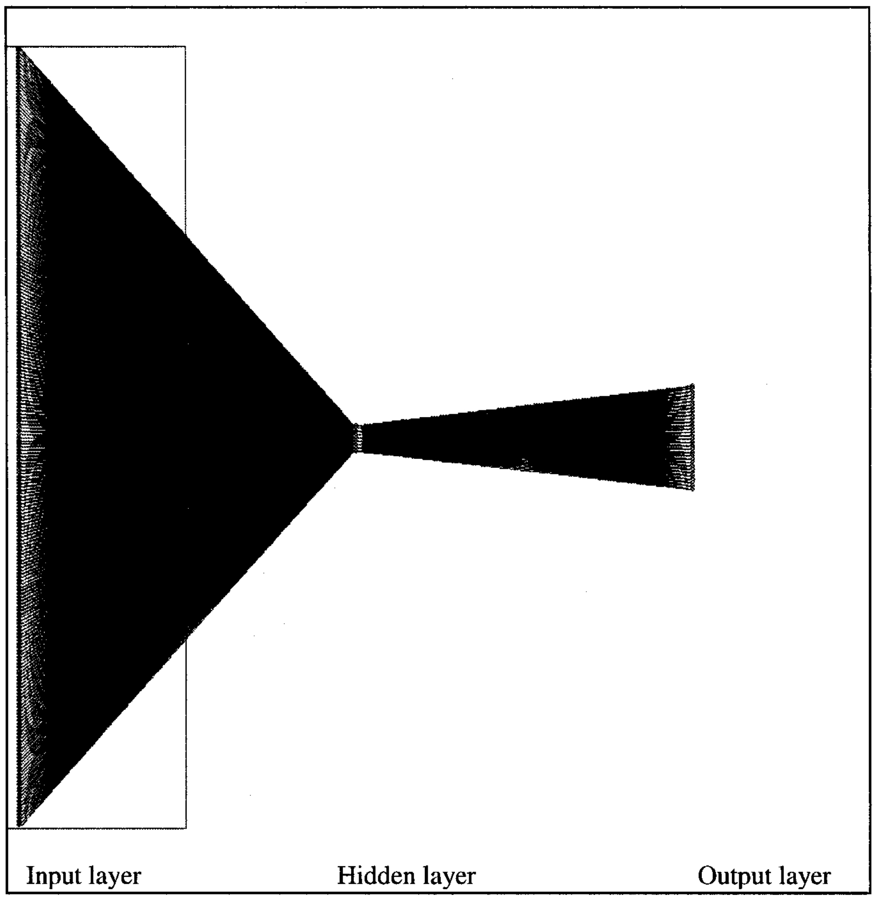


Figure 3.8. Optimum Architecture of the Neural Network for Selection of Machining Operation Sequences

With the above neural network architecture and network internal parameters, the training was performed until the error reached 0.5%. The number of iterations needed was 18471 and the time taken to complete the training was about 16 minutes on a Pentium 4, 1.7 GHz PC with 1GB RAM. The error graph of the training cycle is given in Figure 3.9.

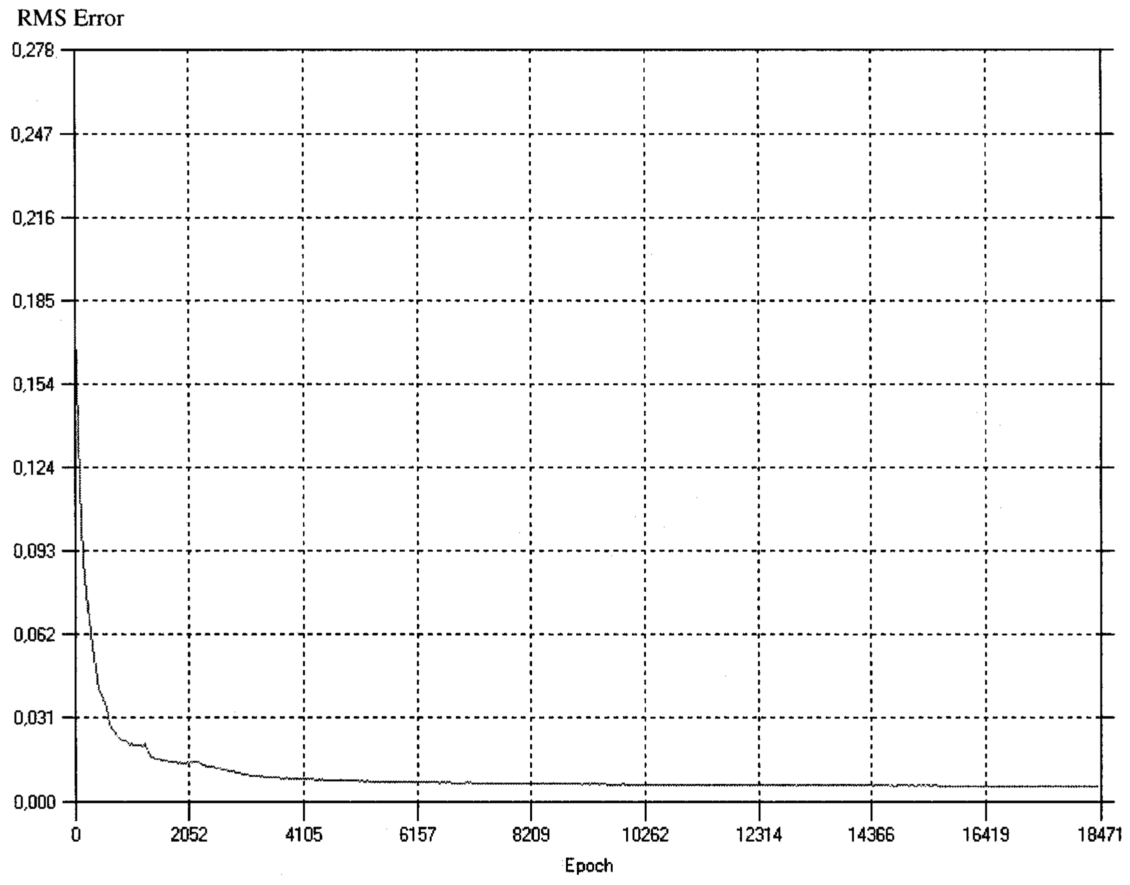


Figure 3.9. Error Graph of the Training Cycle

3.1.4.4 Validation of the Neural Network

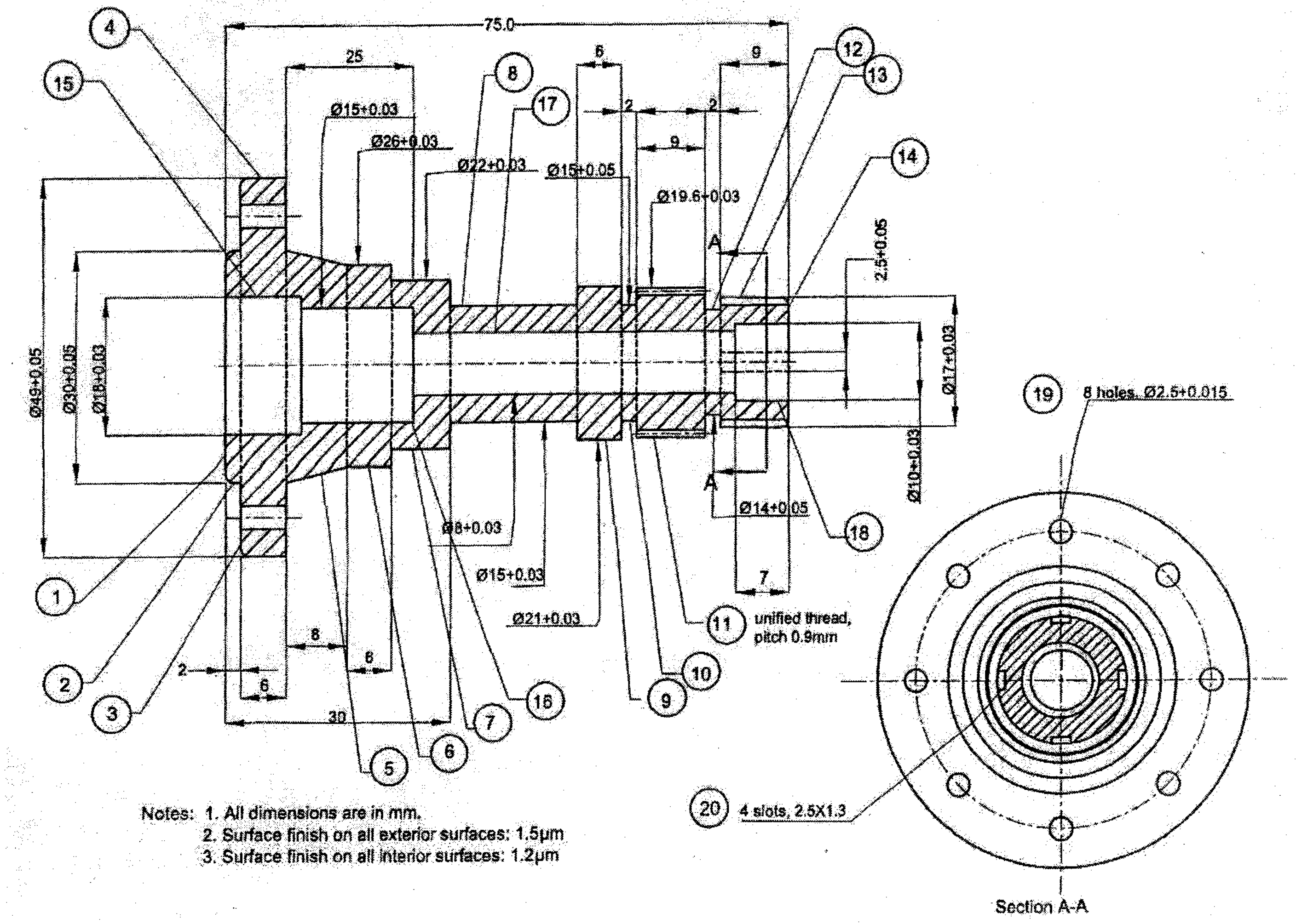
On completion of the training of the neural network, the performance of the trained network has been tested on several sets of input feature attributes, which have not been used as part of the training dataset, in order to examine the validity of the neural network model in selecting machining operation sequences. The results showed a good correlation with the machining data handbook's recommended operation sequences. The purpose of the network validation was to ensure that the developed neural network has the ability to generalize within the limits, set by the training dataset, rather than simply having memorized the input–output relationships that are contained in the training patterns.

3.2. Illustrative Example and Results

In this section, two illustrative examples are used to demonstrate the application of the proposed methodology for selection of machining operation sequences.

3.2.1 Illustrative Example 1

Figure 3.10(a) shows a shaft, for which the machining operation sequences have to be determined. Apart from the slots, all other features are rotationally symmetrical. The part contains the following 30 machining features: features 1, 3, 14 of the type face, features 2, 4, 6, 7, 8, 9, 13 of the type external step, feature 5 of the type external taper, features 10, 12 of the type groove, feature 11 of the type external thread, features 15, 16, 17, 18 of the type hole, feature 19 - 8 numbers of the type hole and feature 20 - 4 numbers of the type slot. Table 3.5 shows the types of features along with the values of attributes of the features namely diameter/width, tolerance and surface finish. The above information together constitutes the input to the neural network. Next the above input data is represented in the input data format of the neural network and presented to the input layer of the neural network. Table 3.6 summarises the results of the output automatically generated by the neural network that includes all the possible machining operation sequences for producing each feature of the part. The above results exhibit a good correlation with the Machining Data Handbook's recommended operation sequences. In the present work, the selection of machining operations has been done one feature at a time including within-the-feature process sequencing. The machining operation sequences for producing each feature of the part, as shown in Table 3.6, will then form the input for the set-up planning module. This module, along with other information such as the tool access directions and the geometric tolerance relationships between the features, will further determine the groups of machining operation sequences that can be performed together, the sequence of machining operations across all the features within the same set-up and the details of locating and clamping the part in each set-up. It is to be noted that for the features 15, 16 and 18, deep hole drilling is not necessary as the length to diameter ratio is well within 3, while for the feature 17, the deep hole drilling alternative has to be chosen, since the length to diameter ratio exceeds 3. It took less than 5 seconds on a Pentium 4, 1.7 GHz PC with 1GB RAM to generate all the above operation sequences. The selected component in the example problem represents the complexity encountered in today's sophisticated manufacturing industries such as aircraft engine production.



Notes: 1. All dimensions are in mm.
2. Surface finish on all exterior surfaces: 1.5μm
3. Surface finish on all interior surfaces: 1.2μm

Figure 3.10(a). A rotationally symmetrical part

Table 3.5. Different Features of the Part Shown in Figure 3.10(a) and Their Attributes

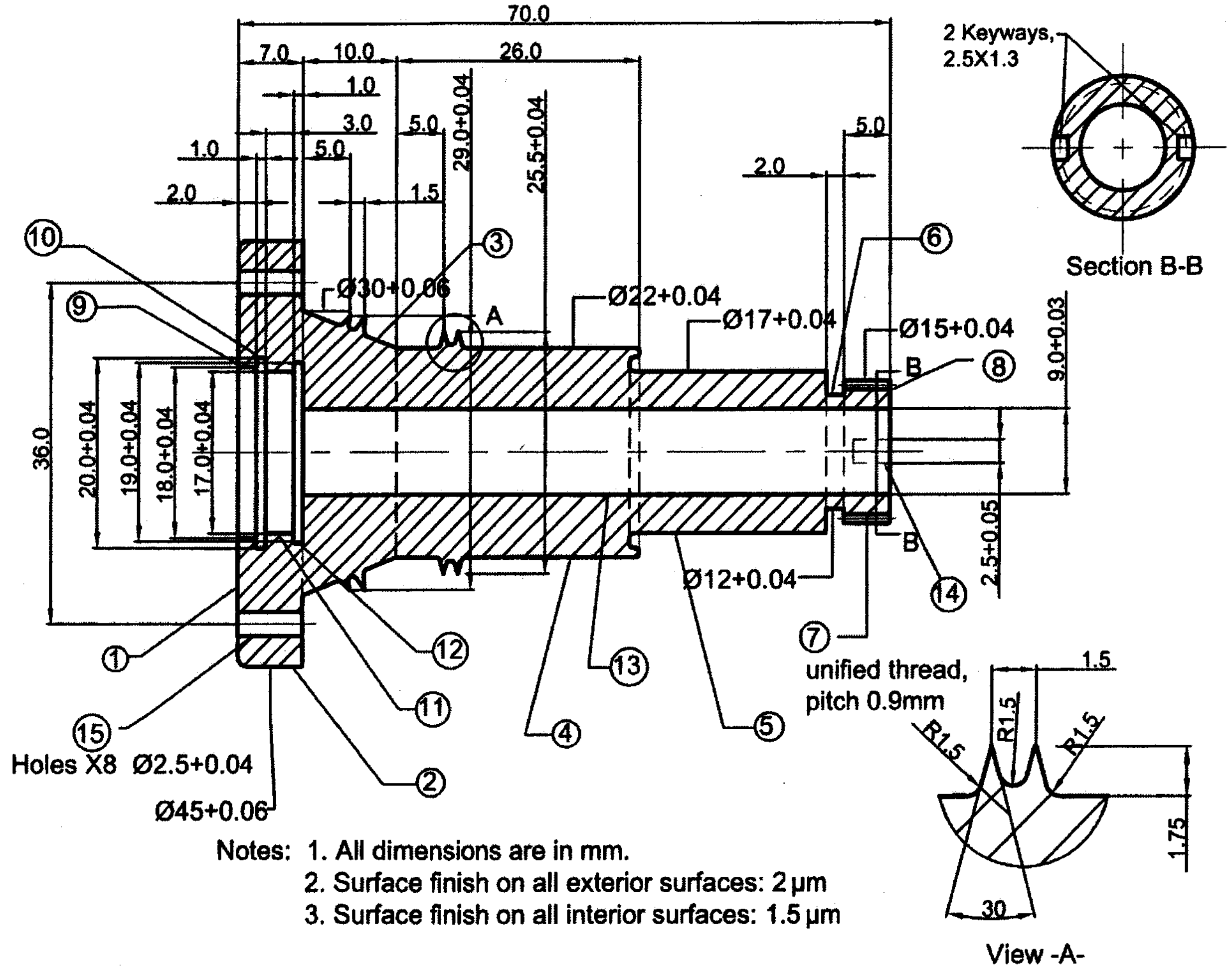
Feature identifier	Type	Diameter/ Width mm	Tolerance mm	Surface finish µm
1	Face	30	0.05	1.5
2	External step	30	0.05	1.5
3	Face	49	0.05	1.5
4	External step	49	0.05	1.5
5	External taper	30 (max dia)	0.05	1.5
		26 (min dia)	0.03	
6	External step	26	0.03	1.5
7	External step	22	0.03	1.5
8	External step	15	0.03	1.5
9	External step	21	0.03	1.5
10	Groove	15	0.05	1.5
11	External thread	19.6	0.03	1.5
12	Groove	14	0.05	1.5
13	External step	17	0.03	1.5
14	Face	17	0.03	1.5
15	Hole	18	0.03	1.2
16	Hole	15	0.03	1.2
17	Hole	8	0.03	1.2
18	Hole	10	0.03	1.2
19	Hole	2.5	0.015	1.2
20	Slot	2.5 (width)	0.03	1.5

Table 3.6. Machining Operation Sequences Generated by the Neural Network for Producing Different Features of the Part Shown in Figure 3.10(a) (Alt. Stands for Alternative)

Feature identifier	Feature type	Operation sequences generated by the neural network
1,3,14	Face	Rough turn→ Semi finish turn→ Finish turn
2,4, 6,7,8,9,13	External step	Alt. 1: Rough Turn→ Semi finish Turn→ Finish Turn Alt. 2: Rough Turn→ Semi finish Turn→ Rough Grind
5	External taper	Alt. 1: Rough Turn→ Semi finish Turn→ Finish Turn Alt. 2: Rough Turn→ Semi finish Turn→ Rough Grind
10,12	Groove	Groove turning (two passes)
11	External thread	Rough Turn→ Semi finish Turn→ Finish Turn→ Threading
15,16,17,18	Hole	Alt. 1: Drill→ Rough Bore→ Semi finish Bore→ Finish Bore; Alt. 2: Deep hole drill
19	Holes X 8 nos.	Alt. 1: Drill-Rough Bore-Semi finish Bore-Finish Bore ; Alt. 2: Deep hole drill
20	Slot X 4 nos.	Rough mill-Semi finish mill

3.2.2 Illustrative Example 2

Figure 3.10(b) shows the example of another shaft, for which the machining operation sequences have to be determined. All the features present in the part, apart from the keyway are rotationally symmetrical. The part contains the following 15 machining features: features 1, 8 of the type face, features 2, 4, 5 of the type external step, feature 3 of the type external taper, feature 6 of the type groove, feature 7 of the type external thread and the features 9, 10, 11, 12 and 13 of the type hole, feature 14 (2 in number) of the type keyway and feature 15 (8 in number) of the type hole. Table 3.7 shows the types of features along with the values of attributes of the features, namely diameter/width, tolerance and surface finish. The above information, which constitutes the input to the neural network, is represented in the input data format of the neural network and presented to its input layer. Table 3.8 gives the results of the output that has been automatically generated by the neural network that includes all the possible machining operation sequences for producing each feature of the part. The above results demonstrate a good correlation with the operation sequences recommended by the Machining Data Handbook. It is to be noted that for the features 9, 10, 11 and 12, deep hole drilling is not necessary as the length to diameter ratio is well within 3, while for the feature 13, the deep hole drilling alternative has to be chosen, since the length to diameter ratio exceeds 3. It took less than 5 seconds on a Pentium 4, 1.7 GHz PC with 1GB RAM to generate all the above operation sequences.



Notes: 1. All dimensions are in mm.
2. Surface finish on all exterior surfaces: 2 µm
3. Surface finish on all interior surfaces: 1.5 µm

Figure 3.10(b). A rotationally symmetrical part

Table 3.7. Different Features of the Part Shown in Figure 3.10(b) and Their Attributes

Feature identifier	Type	Diameter/ Width mm	Tolerance mm	Surface finish µm
1	Face	45	0.06	2
2	External step	45	0.06	2
3	External taper	30 (max dia)	0.06	2
		22 (min dia)	0.04	
4	External step	22	0.04	2
5	External step	17	0.04	2
6	Groove	12	0.04	2
7	External thread	15	0.04	2
8	Face	15	0.04	2
9	Hole	18	0.04	1.5
10	Hole	20	0.04	1.5
11	Hole	17	0.04	1.5
12	Hole	19	0.04	1.5
13	Hole	9	0.03	1.5
14	Keyway	2.5 (width)	0.05	2
15	Hole	2.5	0.04	1.5

Table 3.8. Machining Operation Sequences Generated by the Neural Network for Producing Different Features of the Part Shown in Figure 3.10(b) (Alt. Stands for Alternative)

Feature identifier	Feature type	Operation sequences generated by the neural network
1,8	Face	Rough turn→ Semi finish turn→ Finish turn
2,4, 5	External step	Alt. 1: Rough Turn→ Semi finish Turn→ Finish Turn Alt. 2: Rough Turn→ Semi finish Turn→ Rough Grind
3	External taper	Alt. 1: Rough Turn→ Semi finish Turn→ Finish Turn Alt. 2: Rough Turn→ Semi finish Turn→ Rough Grind
6	Groove	Groove turning (two passes)
7	External thread	Rough Turn→ Semi finish Turn→ Finish Turn→ Threading
9,10,11,12,13	Hole	Alt. 1: Drill→ Rough Bore→ Semi finish Bore → Finish Bore Alt. 2: Deep hole drill
14	KeywayX 2 nos.	Rough mill-Semi finish mill
15	Holes X 8 nos.	Alt. 1: Drill-Counter Bore

3.3. Discussions

As mentioned in the review of literature, Knapp et al (1992), Devireddy et al (1999, 2002) also used the back-propagation learning based neural network to solve the problem of selection of machining operation sequences. However, there are three major differences as compared to the neural network methodology presented in this work with regard to the input representation format, the output representation format and the method of choosing input patterns of the training examples for training the network. The above differences are explained in the following paragraphs along with some of the important advantages that stand to be gained from the approach proposed by the author.

3.3.1 Comparison of the Proposed Neural Network Approach with Other Approaches Developed by Previous Researchers

In the neural network models developed by Knapp et al and Devireddy et al, each input layer node represents an input decision variable and it is assigned the corresponding numerical value of the input decision variable that it represents. In the neural network model that has been developed by the author, the network has been prestructured with domain knowledge gathered in the form of thumb rules prior to training. Each input layer node represents a range of the input decision variable found in the antecedent 'IF' part of the thumb rules and each node is assigned a binary value of either 1 or 0 to indicate whether a certain input decision variable falls within the range represented by the node or not. A method has been presented, based on simple classification rules, to classify the value of each input decision variable and assign a binary value to each input layer node.

The neural network models of Knapp et al and Devireddy et al recommend a single machining operation sequence for a given feature of the part; each output layer node represents a single operation and is assigned a binary value of either 1 or 0 depending on whether a certain machining operation is included in the operation sequence for machining the feature or not. The neural network model developed by the author recommends all the possible operation sequences for machining a given feature of the part; each output layer node, in this case, represents a possible sequence of machining operations and is assigned a binary value of 1 or 0 depending on whether it is suitable for machining the feature or not.

In the neural network approach discussed in the articles above, there are no guidelines on how to choose input patterns of the training examples for the neural network. While choosing the input patterns, the values of input variables are at first randomly selected from the entire range of values possible and an initial training dataset is generated. With this generated training dataset, the neural network is trained. Then the generalization capability of the neural network is verified by validation tests by presenting, to the neural network, examples (validation dataset) describing intermediate situations with respect to those proposed during the training. If the network fails the validation test, the training dataset has to be modified by adding, to the training dataset, the situations of the validation dataset, which have generated greater errors. This process is repeated until the neural network is able to sufficiently generalize within the limits, set by the training dataset. In the present approach by the author, a more systematic method of choosing input patterns for the training examples has been illustrated. The thumb rules developed for selection of machining operation sequences are used to serve as guidelines during the preparation of training examples. The input patterns of training examples have to be chosen in such a way that they activate one or more of these thumb rules. It is done by selecting the input variable values that fall in the ranges related to these rules rather than randomly selecting the values of input variables from their entire range. The neural network is then trained and validated in manner similar to the one described above. This method of choosing input patterns of training examples is found to help ensuring in a better way that the entire problem domain is represented and requires substantially lesser number of training examples.

For process planning of complex components similar to that shown in Figure 3.10, the method adopted by the author offers a generative approach to CAPP that can help to overcome a number of limitations of the variant approach. First of all, unlike the variant approach, the method presented is not just limited to similar parts previously planned. Moreover it is able to generate process plans automatically and consistently over and over again. By eliminating the need for expert process planners while generating the process plan, the present method can also significantly cut down the time and hence the costs in process planning and ensures that the process plan is free from sources of human error that may otherwise occur with the variant CAPP approach. In the event of any major changes in product design or the shop floor manufacturing environment, the method presented may be quickly adapted to accommodate the above changes by necessary modifications of the knowledge base through mere retraining of the network.

The present method here also overcomes various limitations of other generative CAPP approaches such as decision trees and expert systems by its ability to automatically acquire knowledge and generalize beyond the set of training examples. Both the decision trees and the expert systems are relatively static in terms of representing the process planning knowledge. In the proposed method by the author, modification of the process planning knowledge base can be accomplished relatively more easily and quickly by merely retraining the network with the new set of training examples. Another advantage that the present method offers is that, as compared to the decision trees or expert systems, it leads to faster inference, owing to the parallel architecture of the network. The results of the tests carried out by the author on a variety of example parts show that the computation time for generating machining operation sequences by the neural network model developed by the author is reasonably fast so that it can be used to quickly solve most practical problems. This translates directly to the planning time saved in the process planning stage and hence reduces the cost.

3.3.2 Flexibility and Adaptability of the Developed Neural Network Model

The illustrative examples have demonstrated the potential for application of the developed neural network based methodology for machining operations selection of a wide variety of rotationally symmetrical parts. The types of the features that have been considered are internal surfaces of revolution like holes, external surfaces of revolution like external diameters, external taper, external thread, groove, face, and other features such as slot, keyway, and so on, which are commonly encountered in rotationally symmetrical parts. The proposed method is characterized by its ability to automatically acquire knowledge by learning from training examples and to generalize beyond the set of training examples within the limits set by the training dataset as will be shown later. Further it has a flexible architecture and it is characterized by its ability to adapt itself readily to changes in the manufacturing environment as will be explained below.

In the neural network based methodology proposed by the author, modification of the process planning knowledge base can be accomplished relatively more easily and quickly by merely retraining the network with the new set of training examples. In future, for example, in the light of new developments in the manufacturing technology, if a new process is developed or a more optimal sequence is found, the knowledge base will have to be modified. In order to accomplish this, first the new knowledge will have to be expressed in the form of thumb rules. Then the antecedent and consequent parts of the new thumb rules will have to be scanned and if a new range of one of the input decision variables or operation sequence is found, a new node must be created in the input or output layer of the network. If the new thumb rules are not in direct conflict with the pre-existing rules set in the knowledge base, then it might be sufficient to just add to the previous training dataset, a new set of training examples so as to activate the new rules and retrain the network. If, however, one or more pre-existing thumb rules becomes irrelevant as a result, they have to be identified and the nodes representing the range of input decision variables and operation sequences corresponding to the affected rules will have to be removed from the input and output layers. Also the training examples meant for activating these thumb rules will have to be removed from the previous training dataset and the neural network will have to be retrained. The following gives an example of how the above may be accomplished.

Suppose that the honing operation for finishing a hole feature was not initially included in the list of operations considered in the knowledge base of the neural network. To include the honing operation in the knowledge base of the neural network, first the process capability knowledge of the honing process will have to be expressed in the form of thumb rules like the ones shown below.

Thumb rule 1:
IF (Feature is of the type Hole) AND (Dimension of the Hole is Dim_1) AND ... (Tolerance of the Feature is Tol_1) AND (Surface finish of the Feature is SF_1), THEN (Operation sequence is Drilling-Rough Boring-Semifinish Boring-Grinding-Honing)

Thumb rule 2:
IF (Feature is of the type Hole) AND (Dimension of the Hole is Dim_2) AND ... (Tolerance of the Feature is Tol_2) AND (Surface finish of the Feature is SF_2), THEN (Operation sequence is Drilling-Rough Boring-Semifinish Boring-Grinding-Honing)
and so on.

Then the antecedent (IF) part and the consequent (THEN) part of the above thumb rules will have to be scanned to identify all the new ranges of input decision variables like dimension, tolerance and surface finish and output decision variables, which, in the present case, are new operation sequences. Then an additional neuronal node will have to be created in the input layer of the neural network to represent each of the above new dimension ranges namely Dim_1, Dim_2, tolerance ranges namely Tol_1 Tol_2 and surface finish ranges namely SF_1 SF_2. Further an additional neuronal node will have to be created in the output layer of the neural network to represent the new operation sequence, namely Drilling-Rough Boring-Semifinish Boring-Grinding-Honing. Next the input-output pair of training examples will have to be prepared so as to activate the above thumb rules and added to the training dataset of the neural network. The neural network is then retrained and validated in the same manner as already described earlier.

3.3.3 Generalization Capability of the Neural Network Model

The neural network model has the ability to generalize within the limits set by the training dataset. The Figure 3.11 shows the plot of the expected outputs versus the actual outputs generated by the neural network model developed for machining operations selection. Table 3.9 summaries the input data to the neural network used to plot the above outputs. The triangular data points represent the expected outputs and the circular data points represent the actual outputs generated by the neural network model. The input data to the neural network used to generate the above outputs have been adopted from the two illustrative examples shown earlier and carefully chosen in such a way that they are different from the data, which was used earlier for constructing the training dataset of the neural network. From the Figure 3.11, it can seen that the actual outputs generated by the neural network are very close to the expected outputs, which indicates that the neural network model exhibits good generalization capability within the limits set by the training dataset, and that it has not just memorized the input–output relationships that are contained in the training patterns.

Table 3.9. Input Data to the Neural Network Used to Plot the Outputs in Figure 2

Sample no.	Input to the neural network				Outputs	Values of the Outputs	
	Feature type	Dia/ width	Tolerance	Surface finish	Machining operation sequence	Expected Output	Actual Output
1	4	30	50	1.5	Rough turn→ Semi finish turn→ Finish turn	1	0.991483
2	2	30	50	1.5	Rough Turn→ Semi finish Turn→ Finish Turn	1	0.992373
3	4	49	50	1.5	Rough turn→ Semi finish turn→ Finish turn	1	0.991483
4	2	49	50	1.5	Rough Turn→ Semi finish Turn→ Finish Turn	1	0.992373
5	6	30	30	1.5	Rough Turn→ Semi finish Turn→ Finish Turn	1	0.994585
6	2	26	30	1.5	Rough Turn→ Semi finish Turn→ Finish Turn	1	0.992243
7	2	22	30	1.5	Rough Turn→ Semi finish Turn→ Finish Turn	1	0.992243
8	2	15	30	1.5	Rough Turn→ Semi finish Turn→ Finish Turn	1	0.991397
9	2	21	30	1.5	Rough Turn→ Semi finish Turn→ Finish Turn	1	0.992243
10	3	15	50	1.5	Groove turning (two passes)	1	0.975232
11	7	19.6	30	1.5	Rough Turn→ Semi finish Turn→ Finish Turn→ Threading	1	0.991343
12	3	14	50	1.5	Groove turning (two passes)	1	0.975232
13	2	17	30	1.5	Rough Turn→ Semi finish Turn→ Finish Turn	1	0.991397
14	4	17	30	1.5	Rough turn→ Semi finish turn→ Finish turn	1	0.994381
15	1	18	30	1.2	Drill→ Rough Bore→ Semi finish Bore→ Finish Bore	1	0.99195
16	1	15	30	1.2	Drill→ Rough Bore→ Semi finish Bore→ Finish Bore	1	0.990657
17	1	8	30	1.2	Drill→ Rough Bore→ Semi finish Bore→ Finish Bore	1	0.991333
18	1	10	30	1.2	Deep hole drill	1	0.990657
19	1	2.5	15	1.2	Drill→ Rough Bore→ Semi finish Bore→ Finish Bore	1	0.992921
20	5	2.5	30	1.5	Rough mill→ Semi finish mill	1	0.982788

Sample no.	Input to the neural network				Outputs	Values of the Outputs	
	Feature type	Dia/ width	Tolerance		Machining operation sequence	Expected Output	Actual Output
21	4	45	60	2	Rough turn→ Semi finish turn→ Finish turn	1	0.991483
22	2	45	60	2	Rough turn→ Semi finish turn→ Finish turn	1	0.991047
23	6	30	40	2	Rough turn→ Semi finish turn→ Finish turn	1	0.990378
24	2	22	40	2	Rough turn→ Semi finish turn→ Finish turn	1	0.990233
25	2	17	40	2	Rough turn→ Semi finish turn→ Finish turn	1	0.990993
26	3	12	40	2	Groove turning (two passes)	1	0.975232
27	7	15	40	2	Rough Turn→ Semi finish Turn→ Finish Turn→ Threading	1	0.991839
28	4	15	40	2	Rough turn→ Semi finish turn→ Finish turn	1	0.994381
29	1	18	40	1.5	Drill→ Rough Bore→ Semi finish Bore→ Finish Bore	1	0.991129
30	1	20	40	1.5	Drill→ Rough Bore→ Semi finish Bore→ Finish Bore	1	0.991129
31	1	17	40	1.5	Drill→ Rough Bore→ Semi finish Bore→ Finish Bore	1	0.993773
32	1	19	40	1.5	Drill→ Rough Bore→ Semi finish Bore→ Finish Bore	1	0.991129
33	1	9	30	1.5	Deep hole drill	1	0.995148
34	5	2.5	50	2	Rough mill→ Semi finish mill	1	0.982789
35	1	2.5	40	1.5	Drill→ Counterbore	1	0.997937

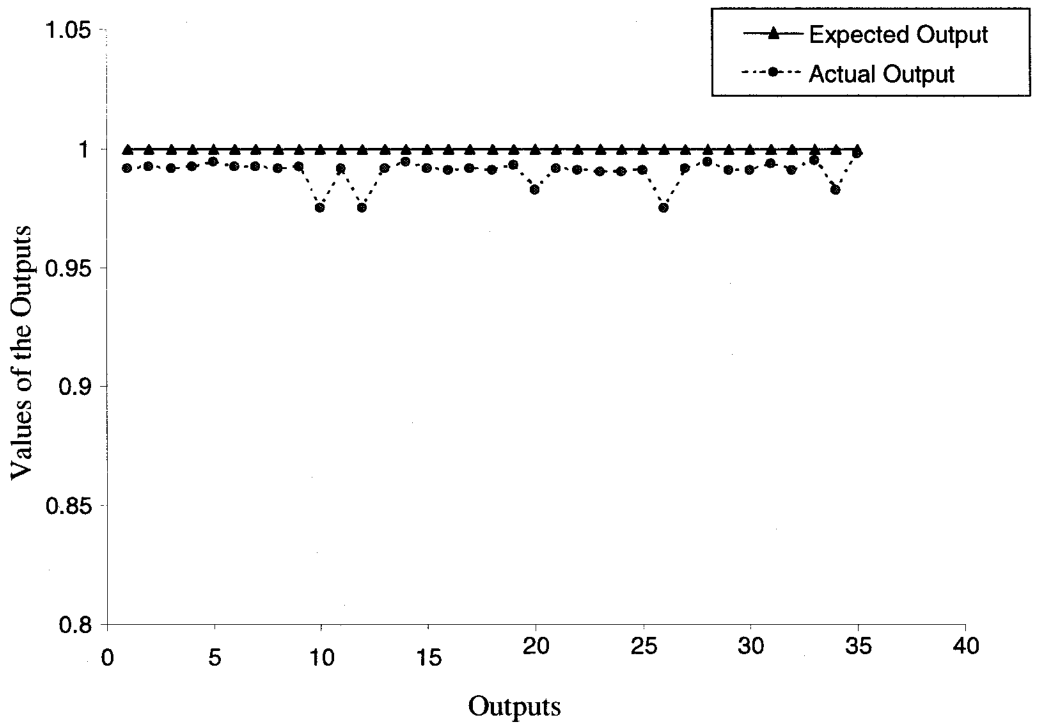


Figure 3.11. Plot of the Expected Outputs versus the Actual Outputs Generated by the Neural Network

3.3.4 Ability of the Neural Network Model to Deal with Incomplete or Erroneous Input Data

Another advantage of the neural network models often reported by previous researchers is that it is robust with respect to incomplete or erroneous input data e.g. data involving some missing or erroneous values of the input feature attributes for the present machining operations selection problem. The decision trees and expert systems are incapable of dealing with such incomplete or erroneous input data. This can be explained from the fact that the neural network has the ability to learn, from the training examples, subtle relationships between the input and output, if they exist and retrieve regularities from the data flow. So even if the input data presented to it is incomplete or erroneous, the neural network model will still function by retrieving the relationship between the input data and the output data from the relationships known to it, to generate reasonable mappings and recommend the correct outputs possible. This is particularly useful in the case of problems of machining operations selection, where a number of input decision variables are involved. However, it is not within the scope of the present research work to verify it, since only limited number of input decision variables have been considered for the present problem of machining operations selection. But this could be a direction for future research.

3.3.5 Integration of the Network Network Program for Machining Operations Selection with Other CAPP Submodules

Also it is important to note that although in the present work, the neural network for machining operations selection has been developed as a stand alone application program, it is possible to extract the source code of the program in C and then embed it within any user defined C application program. This feature enables the neural network module for machining operations selection to be integrated with other modules of the

CAPP system such as module for set-up planning and module for automatic feature extraction from CAD system. From the above discussions, it is evident that the neural network based approach to machining operations selection developed by the author has lot of potential to be used as an alternative to the traditional variant and generative CAPP approaches.

CHAPTER 4

PROPOSED EXPERT SYSTEM BASED METHODOLOGY FOR SET-UP PLANNING

The expert system based methodology developed by the author for set-up planning has been described in detail in the present chapter. It caters to the process planning of machined rotationally symmetrical parts containing different features such as internal surfaces of revolution like holes, external surfaces of revolution like external diameters, external taper, external thread, groove, face, and other features such as slot, keyway, and further parts that may contain overlapping features, e.g. an external thread and an overlapping keyway and so on. It has been implemented on a PC by using the rule based expert system shell, CLIPS Version 6.10 (Giarratano et al 1998). It takes in, as input, information about different features present in the part to be machined along with the operations required for machining each feature and is capable of performing automatically the different set-up planning tasks such as set-up formation, operation sequencing and selection of locating and clamping surfaces for each set-up. The key issues regarding the expert system based methodology that have been presented here include development of the overall structure of the expert system, the database, the rule based knowledge base for solving the different problems of set-up planning and the inference engine and how to accomplish modification of the expert systems knowledge base so as to adapt it to new situations. The examples of some industrially-relevant rotationally symmetrical parts have been analysed using the proposed approach to demonstrate its potential for application in the real manufacturing environment and the results and discussions have been presented.

4.1 Development of the Overall Structure of the Expert System

The overall structure of the expert system that has been developed is shown in Figure 4.1. It consists of a database, a rule based knowledge base, an inference engine, a user interface and an explanation facility, details of which are given below.

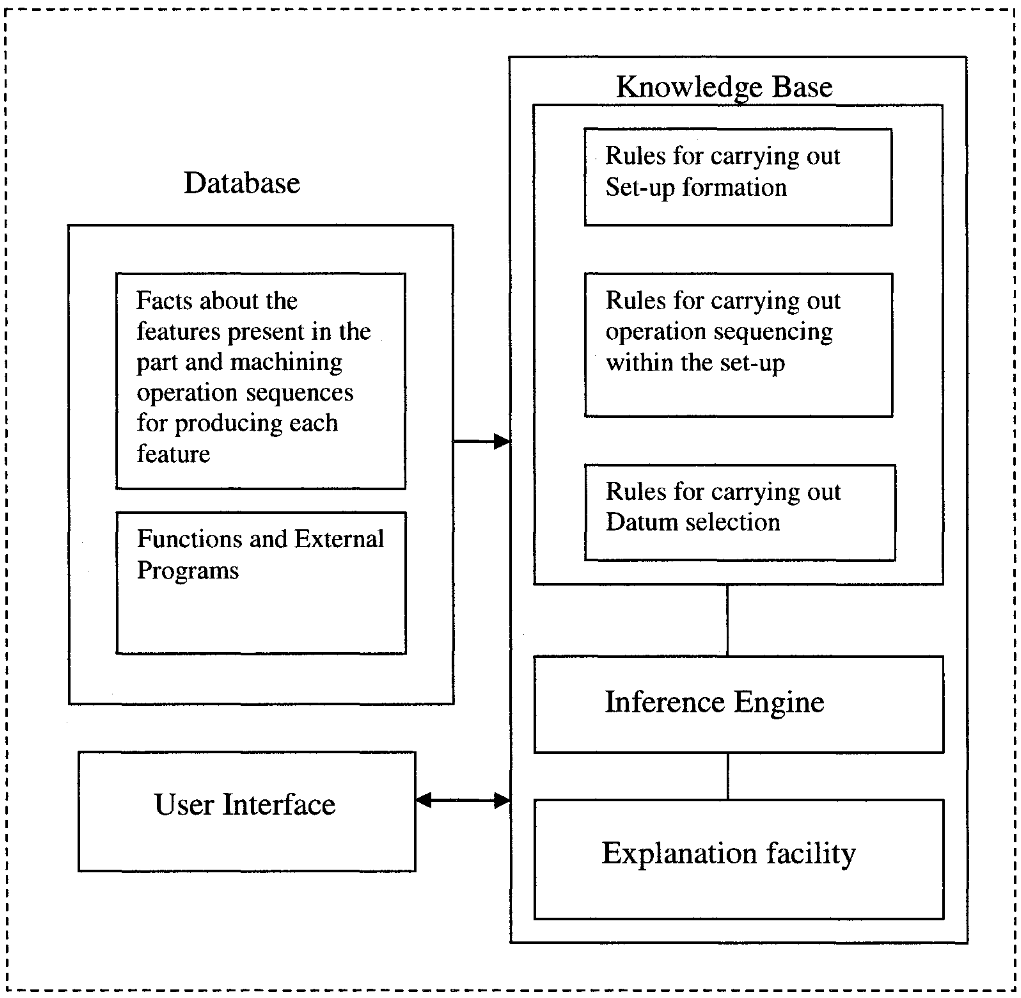

Database
Facts about the features present in the part and machining operation sequences for producing each feature
Functions and External Programs
User Interface
Knowledge Base
Rules for carrying out Set-up formation
Rules for carrying out operation sequencing within the set-up
Rules for carrying out Datum selection
Inference Engine
Explanation facility

Figure 4.1. Overall Structure of the Expert System for Set-up Planning

4.1.1 Database

The database contains the declarative knowledge of the expert system that includes the detailed information about the different features present in the part and the operations required for machining each feature, organised and presented in the form of data files. They constitute the input data for the expert system for set-up planning. The following section explains the format of representation that has been developed for the above input data. In addition to the data files, the database also contains functions and external programs that are necessary for performing certain calculations.

4.1.1.1 Data files

The input information necessary for set-up planning includes the type of features present in the part, their dimensions, the geometric tolerance relationship (if any) with respect to other features, and the tool access directions (TAD's) in order to machine each feature. These information need to be extracted from the original part design on the CAD system. It also includes the different machining operations needed to produce each feature, which need to be obtained from the process selection module. The following explains the format of representation developed for the above input data. The commonly used input data representation format in CLIPS is the template. The concept of template is somewhat analogous to a record structure in a programming language such as Pascal. It is a list of several named fields called slots which, in turn, may contain one or more fields. Each slot stores values (a single value in the case of a slot and zero or more values in case of a multislot), which can be restricted to a certain range or selected from a set of predefined values. A format for representation of the input data about features has been developed using the template as shown in Figure 4.2(a). It has one slot for each of the following namely, the feature identifier, name of the feature (e.g. external step, hole, etc.), type of feature (internal or external), sub-type of feature (primary or secondary), the feature to which it is secondary, adjacent feature identifiers, names of all the adjacent features, identifiers of the reference features, feature diameter, the adjacent feature diameters and the Tool Access Direction (left, right or both) in order to machine the feature. Using the above template definition for feature, the input data on a typical feature may be entered as follows:

(feature (number 4) (name EXTERNAL_STEP) (type EXTERNAL) (subtype PRIMARY) (adjacent_features 3 5) (adjacent_features_names FACE EXTERNAL_TAPER) (step_diameter 49) (TAD right-left))

A format for representation of the input data about machining operations has been developed using the template as shown in Figure 4.2(b). It has one slot for each of the following, namely the operation identifier, type of operation, details on stage of machining (rough, semi-finish or finish), feature identifier on which it acts, the TAD of the concerned feature, list of features with which it has a tolerance relationship and the respective tolerance values in μm. Using the above template definition for operation, the input data on a typical machining operation may be entered by the user as follows:

(operation (number 401) (type turn) (machining_stage rough) (on-feature 4) (TAD right-left) (relation-with-feature 2 13) (tolerance 0.1 0.2))

The reason for choosing the template for representation of the input data, as clearly evident from the above discussion, is to make the facts more readable and easier to work with. There are two ways by which the above input data can be entered into the developed expert system. The input data can be saved as a data file with the extension .clp and loaded from the file into the expert system environment at the time of execution. Alternatively, it may be also directly entered manually by typing through a user interface.

```
(deftemplate MAIN::feature
(slot number (type INTEGER) (default ?NONE))
(slot name (type SYMBOL) (allowed-symbols EXTERNAL_STEP FACE GROOVE HOLE KEYWAY
EXTERNAL_TAPER THREAD HOLE))
(slot type (type SYMBOL) (allowed-symbols EXTERNAL INTERNAL))
(slot subtype (type SYMBOL) (allowed-symbols PRIMARY SECONDARY))
(slot secondary_feature_to (type INTEGER) (default ?DERIVE))
(multislot adjacent_features (type INTEGER) (default ?DERIVE))
(multislot adjacent_features_names (type SYMBOL) (allowed-symbols EXTERNAL_STEP FACE
GROOVE HOLE KEYWAY EXTERNAL_TAPER THREAD HOLE))
(multislot reference_features (type INTEGER) (default 0))
 (slot step_diameter (type NUMBER))
(multislot adjacent_step_diameters (type NUMBER))
(slot hole_diameter (type NUMBER))
(slot hole_depth (type NUMBER))
(multislot adjacent_hole_diameters (type NUMBER))
(multislot adjacent_hole_depths (type NUMBER))
(slot TAD (type SYMBOL) (allowed-symbols left right right-left) (default ?NONE)))
```

(a) Feature Template

```
 (deftemplate operation
(slot number (type INTEGER) (default ?NONE))
(slot type (type SYMBOL) (default ?NONE))
(slot machining_stage (type SYMBOL) (allowed-symbols rough semifinish finish) (default rough))
(slot on-feature (type INTEGER) (default ?NONE))
(slot TAD (type SYMBOL) (allowed-symbols left right right-left) (default ?NONE))
(multislot relation-with-feature (type NUMBER) (default 0))
(multislot tolerance (type NUMBER) (default ?DERIVE)))
```

(b) Operation Template

Figure 4.2 Format of Representation of the Input Data

4.1.1.2 Functions and External Programs

Additionally, functions have been included in the database of the expert system for performing various calculations, such as for example, finding the most critical tolerance relationship of a given feature with respect to other features and other tasks such as, updating the tolerance relationship vectors by removing those tolerance relationships between features that have been already satisfied after assigning the corresponding machining operations to the same set-up.

4.1.2 Knowledge Base

The knowledge base contains the problem solving knowledge of the expert system. It is the collection of rules to be used by the expert system for solving the different set-up planning problems. A knowledge base has been developed for solving each of the problems of set-up formation, operation sequencing and selection of locating and clamping datums, details of which are given in the following sections. The necessary knowledge for formulating the rules has been based on the heuristic knowledge and expert knowledge from various sources such as handbooks and textbooks on machining [Bralla (1986), Halevi et al (1995) and Wang et al (1991)] and based on interviews with experts in process planning as well as discussion with skilled machinists and direct observations of actual machining of workpieces in the shop floor.

4.1.2.1 Knowledge Base for Solving the Set-up Formation Problem Based on Rules

A set of rules have been developed which are capable of clustering the given machining operations for a given machine tool into two groups or set-ups: machining operations, to be performed only from the right and those only from the left, after considering TADs of the corresponding features and the relative tolerance relationships between them. A total of about 14 rules has been developed and included in the knowledge base for solving the problem of set-up formation. The following are examples of some rules that have been included as part of the above knowledge base.

For example, if a machining operation on a feature is encountered in the facts list, which has both TAD's (left and right) but no specified tolerance relationship with any other feature, then it is assigned to any one of the setups from the left or from the right. If a machining operation on a feature is encountered, which has both TADs and further has tolerance relationship with a feature, B having a single TAD, then it is assigned to the same set-up as B. If a machining operation on a feature is encountered, which has both TADs and further has tolerance relationships with more than one feature having a single TAD, then a function (as explained in section 4.1.1.2) is used to first identify the tightest tolerance relationship among them. Then the operation is assigned to the same set-up as the operation on the other feature with which it has the tightest tolerance.

The example of what a typical rule for set-up formation looks like is shown in Figure 4.3. The above rule states that if there exists a fact "operation" about a machining operation on feature A having both TADs and if the slot "tolerance" of the "operation" fact has two or more values i.e. in other words, feature A has tolerance relationship with more than one feature having a single TAD, and if the feature B, with which it has the tightest tolerance, has the TAD "left", then operation on A is also assigned the TAD "left" and it is assigned to the same set-up as the operation on B. The above rule calls three functions. The function "feature-with-tightest-tolerance" analyses the tolerance relationships between A and other features and returns the feature identifier having the tightest tolerance relationship with A. The functions namely, "update-relation-with-feature" and "update-tolerance" updates the "relation-with-feature" and "tolerance" slots of the "operation" fact by removing the feature B from the list of features with which with which A has tolerance relationships because the tolerance relationship between features A and B has been already satisfied.

```
(defrule sample_rule_setup_formation
        ?f1 <- (operation (TAD right-left))
        (test (>= (length$ (fact-slot-value ?f1 tolerance)) 2))
        (operation (TAD left) (on-feature =(feature-with-tightest-tolerance ?f1)))
=>
        (modify ?f1 (TAD left) (relation-with-feature =(update-relation-with-feature ?f1)) (tolerance
=(update-tolerance ?f1))))
```

Figure 4.3. Typical Rule for Set-up Formation

4.1.2.2 Knowledge Base for Solving the Operation Sequencing Problem Based on Rules

The decision on determining the sequence of machining operations within the set-up has been based in accordance with the various precedence constraints imposed by the precedence relationships that may exist between the features and certain manufacturing logic to be followed in ordering the operations that will be discussed below.

A precedence constraint between features, A and B, abbreviated as A→B, implies that B cannot be machined until the machining of A has been completed. A set of rules has been developed, based on heuristic knowledge and expert knowledge from machining text books and handbooks, to derive the precedence constraints between machining operations on features. The following are examples of some rules that have been included as part of the knowledge base for determining the precedence constraints. For example, if a feature, C is the datum or reference for features, A and B, then C has to machined prior to A and B, which will result in precedence constraints. For a rotational part, the end faces that are usually considered the reference features are to be machined first. Some features need to be accessible first before they can be machined, which results in another precedence constraint e.g. in case of machining of a groove between two adjacent external cylindrical surfaces, both the cylindrical surfaces should be machined prior to machining the groove (Figure 4.4a). Another example is that in the case of a chamfer between the face and the cylindrical surface of an external step, both the face and cylindrical surface have to be machined prior to the chamfer (Figure 4.4b). Further there may be certain constraints requiring that the subsequent features should not destroy the properties of features machined previously, which results in another type of precedence. An example is that the machining of a chamfer and a groove must be completed prior to that of the adjacent thread (Figure 4.4c). In case there exists several coaxial holes having the same tool access direction, a good manufacturing practice is to machine the minimum diameter hole first (if its length to diameter ratio is within allowable limits of the tool) followed by the subsequent holes in order of their diameters from small to large, which results in another type of precedence (Figure 4.4d). In case there exist overlapping features such as a thread and an overlapping keyway, a good manufacturing practice is to machine the thread first followed by machining of the keyway, which results in another type of precedence constraint (Figure 4.4e). A total of about 40 rules has been developed and included in the knowledge base for determining the precedence constraints between machining operations on features.

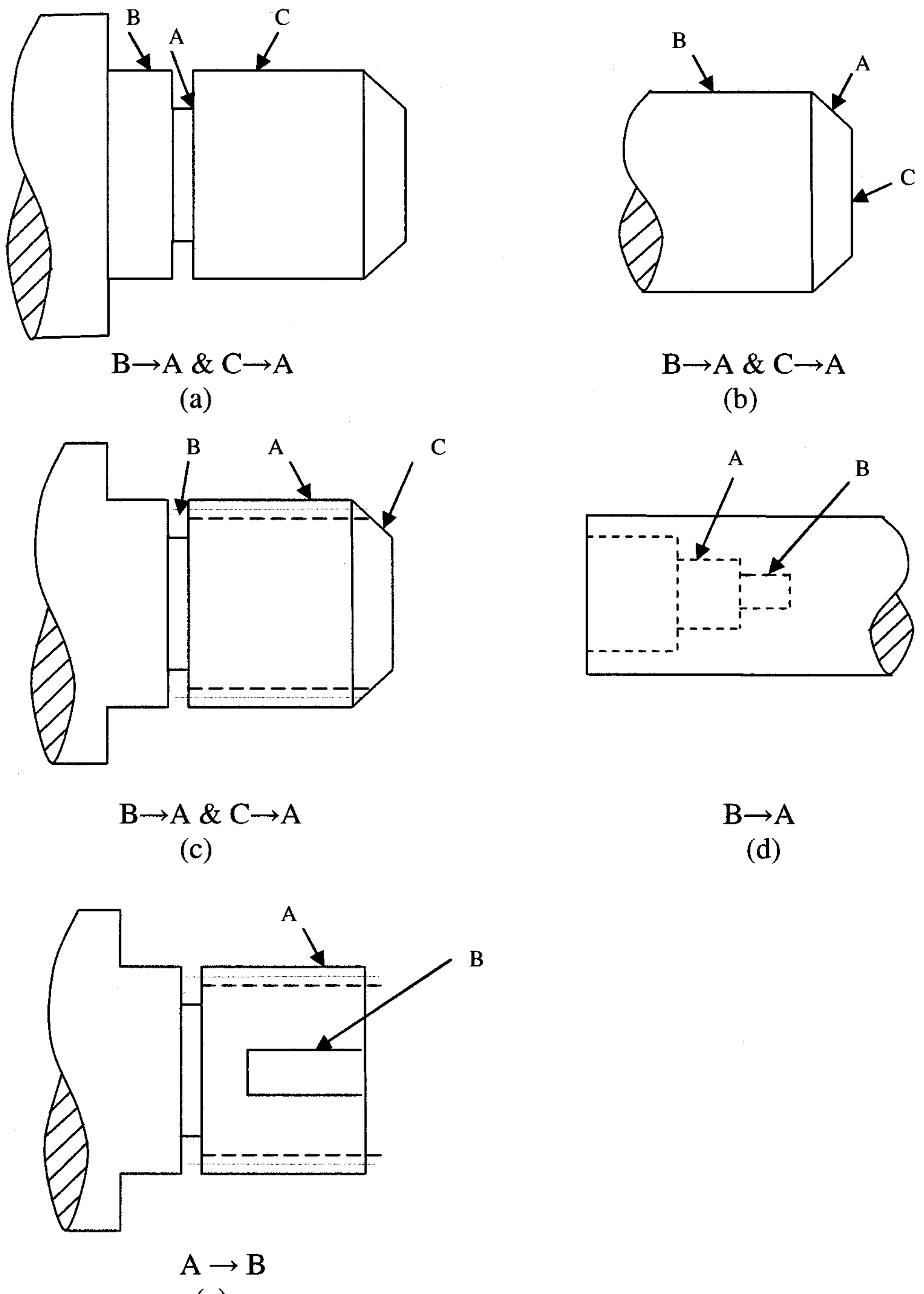


Figure 4.4. Different Kinds of Feature Precedence Relations

Figure 4.5 gives some examples of what typical rules for determining the precedence constraints look like. The sample_rule1 states that if there exists a feature A that is of the type thread having one of the adjacent features, B that is of the type groove, then the precedence between the machining operations on A and B will be first machining of B, followed by machining of A (Figure 4.4c). The sample_rule2 in Figure 4 states that if there exists a feature, A that is of the type hole and of diameter d1 having adjacent features that are also of the type holes of which one of the adjacent features, B has diameter d2 that is smaller than d1, then the precedence between the machining operations on the features A and B will be first machining of B, followed by machining of A (Figure 4.4d).

```
(defrule sample_rule1
        (feature (number ?A) (name THREAD) (adjacent_features $? ?B $?) (adjacent_features_names
$? GROOVE $?))
=>
        (assert (precedence ?B ?A)))

(defrule sample_rule2
        ?f1 <- (feature (number ?A) (name HOLE) (hole_diameter ?d1) (adjacent_features ? ?B)
(adjacent_features_names HOLE HOLE) (adjacent_hole_diameters ? ?d2) (adjacent_hole_depths ? ?h2))
        (test (> ?d1 ?d2))
=>
        (assert (precedence ?B ?A)))
```

Figure 4.5. Typical Rules for Deriving Machining Operation Precedences

For sequencing of operations, in addition to the above precedence constraint information, also two types of manufacturing logic for ordering the various machining operations have been used: first machining of external surfaces, followed by machining of internal surfaces and first rough machining, followed by semi-finish machining, followed by finish machining. The overall priority order for sequencing of machining operations is as follows:

- rough machining of external surfaces along the given tool access direction, followed by

- semi-finish machining of external surfaces along the given tool access direction, followed by
- finish machining of external surfaces along the given tool access direction, followed by
- rough machining of internal surfaces along the given tool access direction with the following priority order, namely drilling followed by rough boring, followed by rough reaming and so on, followed by
- semi-finish machining of internal surfaces along the given tool access direction in the following priority order, namely semi-finish boring followed by semi-finish reaming, followed by counterboring and so on, followed by
- finish machining of internal surfaces along the given tool access direction in the following priority order, namely finish boring followed by finish reaming and so on.

It is to be noted that by grouping all the similar operations together such as for example, grouping all the roughing operations on external surfaces together, it is possible to reduce the number of tool changes and idle tool motions. The above priority order for sequencing of machining operations has been implemented by using a unique feature of CLIPS called salience, which is a mechanism to assign priorities to various rules; thus when multiple rules are present, this feature allows a rule with higher salience to fire before a rule with lower salience. Examples of implementation of priority order for sequencing of machining operations will be given later. Also in addition to the above, another priority for sequencing of machining operations is that for a given set-up, the machining of the features is done starting from one end of the part, while respecting the precedence constraints between the features being machined. This preference in machining helps to reduce the tool travel distances and idle motion of the tool by machining as many adjacent features as possible in a given set-up.

Using the above precedence constraint information and the manufacturing logic, a feasible sequence of machining operations within each set-up is automatically generated as a string of operations arranged in the sequential order in which they are to be performed and then the set-up sequence is determined. The following explains how it has been accomplished. The information about the set-up cluster and preceding operations are incorporated as new slots into the template for the "operation" facts, which is redefined as shown in Figure 4.6. The slot "setup-cluster" is used to indicate the particular set-up (left or right) to which it belongs and the slot "preceding_opn" to indicate the list of preceding operations. The information on set-up cluster to which an operation belongs is obtained using the rules for set-up formation described in section 4.1.2.1 and the information about preceding operations are obtained using the rules for generating precedence constraints mentioned earlier in this section.

```
(deftemplate MAIN::operation
(slot number (type INTEGER) (default ?NONE))
(slot type (type SYMBOL))
(slot machining_stage (type SYMBOL) (allowed-symbols rough semifinish finish) (default rough))
(slot setup-cluster (type SYMBOL) (allowed-symbols left right))
(multislot preceding_opn (type INTEGER) (default 0)))
```

Figure 4.6. Modified Operation Template

The sequencing of operations within the set-up is accomplished using a set of rules. First of all, two multi-field variables are defined namely, the set-up cluster from the left and that from the right. They will contain the ordered set of operations to be performed from the left and from the right respectively, with each field of the variable standing for an operation identifier. Then all the "operation" facts are scanned and a set of rules explained below is used for assigning each operation to one of the two set-up cluster variables in the sequential order in which they must be performed. For example, if a machining operation having no preceding operations is encountered, then it is assigned as the first element of the set-up cluster variable, or it is assigned to the set-up cluster variable following the operation that was last assigned. If a machining operation is encountered that has all of its preceding operations already assigned to the set-up cluster variable, then also it is assigned to it following the operation that was last assigned. All the assignments are done while maintaining the priority order for operations sequencing. The scanning of "operation" facts is continued until all the operations have been assigned to one set-up cluster or the other.

Figure 4.7 gives some examples of what typical rules for determining the sequence of operations within a set-up look like. First, two global variables have been defined namely, "sequence-left-cluster", indicating the set-up cluster from left and "opn-left-cluster" indicating an operation belonging to the left set-up cluster and they have been initialised to 0. The sample rule 1 states that if an operation n1 belonging to the left set-up cluster is encountered and it is meant for rough-machining of an external step, and if it has one preceding operation n2 that also belongs to the left set-up cluster and has been already assigned to the "sequence-left-cluster" variable, then operation n1 may be assigned to the "sequence-left-cluster" variable. The salience or the priority in execution of the rule 1 among other rules present is set as 99. The sample rule 2 is similar to sample rule 1 except that it is meant for semi-finish machining of an external step and its salience set as 79. The priority in execution of the sample rule 1 is higher than that of sample rule 2, signifying that if the conditions for firing both the rules 1 and 2 are satisfied, then the actions of rule 1 are executed first followed by that of rule 2, which, in turn, causes the rough machining operation to be assigned to the operations sequence ahead of the semi-finish machining operation. This is in accordance with the manufacturing logic that rough machining should precede semi-finish machining.

```
(defglobal ?*sequence-left-cluster* = 0
           ?*opn-left-cluster* = 0)

(defrule sample-rule-1

         (declare (salience 99))
         ?f1 <- (opn (number ?n1) (machining_stage rough) (setup-cluster left) (preceding_opn ?n2))
         (operation (number ?n1) (on-feature ?N1))
         (feature (number ?N1) (name EXTERNAL_STEP))
         (test (not (= ?n2 0)))
         (opn (number ?n2) (machining_stage rough) (setup-cluster left))
=>
         (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
         (if (subsetp (create$ ?n2) (create$ ?*sequence-left-cluster*))
           then
           (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))


(defrule sample-rule-2

         (declare (salience 79))
         ?f1 <- (opn (number ?n1) (machining_stage semifinish) (setup-cluster left) (preceding_opn ?n2))
         (operation (number ?n1) (on-feature ?N1))
         (feature (number ?N1) (name EXTERNAL_STEP))
         (test (not (= ?n2 0)))
         (opn (number ?n2) (machining_stage semifinish) (setup-cluster left))
=>
         (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
         (if (subsetp (create$ ?n2) (create$ ?*sequence-left-cluster*))
           then
           (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
```

Figure 4.7. Typical Rules for Operation Sequencing

4.1.2.3 Knowledge Base for Solving the Datum Selection Problem Based on Rules

The decision on determining as to which of the surfaces are suitable for datum selection has been based in accordance with the following principles:

- select as datum that surface of the part, which has an orientation that is different from the surfaces being machined (recall that in case of machining of rotational parts, two orientations are possible namely, orientation from the left and that from the right) and has the tightest tolerance with one of the surfaces to be obtained in the set-up

- in the case, when no tolerance relationship exists between the surfaces of different orientations, select as datum that surface of the part, which has an orientation that is different from the surfaces being machined and has the largest diameter or the longest cylindrical surface.

The above principles for datum selection have been implemented using a set of rules to determine the locating and clamping surfaces for a given set-up.

Figure 4.8 gives some examples of what typical rules for datum selection look like. The sample-rule-1 states that if a feature C encountered in the facts list, is of the type external step and if the TAD for machining C is left and if C has the tightest geometric tolerance relationship with a feature X of the type external step and if the TAD for machining X is right, then the external cylindrical surface of X may be chosen as the clamping surface and the vertical surface of X may be chosen as the locating surface for the left set-up. The sample-rule-2 states that if none of features to be machined in the left set-up has a tolerance relationship with features to be machined in the right set-up, and if a feature A of the type external step is encountered in the facts list and happens to have the largest diameter among all the features present in the part and if the TAD for machining A is right, then the external cylindrical surface of A may be chosen as the clamping surface and the vertical surface of A may be chosen as the locating surface for the left set-up.

```
(defrule sample-rule-1
        (feature (number ?c) (name EXTERNAL_STEP))
        ?f1 <- (operation (on-feature ?c) (TAD left))
        (test (>= (length$ (fact-slot-value ?f1 tolerance)) 2))
        (operation (on-feature =(feature-with-tightest-tolerance ?f1)) (TAD right))

=>

        (assert (datums_selected (setup left) (clamping_surface =(feature-with-tightest-tolerance ?f1))
(locating_surface =(feature-with-tightest-tolerance ?f1)))))


(defrule MAIN::sample-rule-2
        (not (operation (TAD left) (relation-with-feature ~0)))
        (feature (number ?a) (type EXTERNAL) (name EXTERNAL_STEP))
        (feature-with-largest-dia (number ?a))
        (operation (on-feature ?a) (TAD right))

=>
        (assert (datums_selected (setup left) (clamping_surface ?a) (locating_surface ?a))))
```

Figure 4.8. Typical Rules for Datum Selection

Each of the set of rules developed above has been encoded following the syntax of CLIPS and the above rules have been saved as knowledge base files with the extension .clp. At the time of execution of the expert system program, the rules have to be loaded from the knowledge base files into the expert system environment. Alternatively, they may be also directly entered manually by typing through the user interface before execution of the expert system program.

4.1.3 Inference Engine

The inference engine is that part of the CLIPS expert system shell that is already programmed and ready for use. It is used to make inferences by deciding which rules are satisfied by the facts, prioritize the satisfied rules, and execute the rule with the highest priority. The inference engine of CLIPS is based on the forward chaining strategy.

4.1.4 User Interface and Explanation Facility

In addition to the database, the knowledge base and the inference engine, the CLIPS expert system shell also has two other components, namely a user interface and an explanation facility. The user interface is that part of the CLIPS expert system shell which provides the mechanism by which the user interacts with the expert system. The user interface allows the user to perform various tasks, which include, among others, creating and editing the knowledge base using a text editor, saving the knowledge base as one or more text files, loading the knowledge base and the databases into CLIPS, and executing CLIPS. The user interface also provides commands for viewing the current state of the system, tracing execution, adding or removing information and so on. A simple explanation facility is also provided in the CLIPS expert system shell that displays the rationale behind applying (firing) a certain rule. For example, it displays all the facts from the database which satisfies the patterns of a certain rule from the knowledge base. It is particularly helpful in debugging the knowledge base.

By using the methodology for set-up planning presented above, the CLIPS expert system shell has been designed to function as a stand-alone software program that is capable of automatically performing the different set-up planning tasks in machining of rotationally symmetrical parts. It accepts, as input, the data files with the extension .clp, containing the necessary information about the different features present in the part, such as type of each feature, its dimensions, the geometric tolerance relationship with respect to other features, the tool access directions in order to machine it and the different machining operations needed to produce it. The above information has to be presented using the input data representation format that has been developed in section 4.1.1. Before the expert system program can be executed, first of all, the main program file and the files containing the various knowledge base rules developed in sections 4.1.2 have to be loaded into the CLIPS environment by using the user interface, followed by loading the data files containing the input formation about features present in the part and the machining operations to produce the features. Then once the expert system program is executed, it automatically generates, as output, the list of machining operations to be carried out in each set-up, the sequence in which the operations are to be performed and the surfaces on which the part has to be located and clamped for each set-up. After development of the expert system program, it is important to validate its performance on different parts. Accordingly, the performance of the developed expert system program was validated on a variety of different parts. The results showed good correlation and consistency with the set-up plans recommended by human expert process planner for the machining the above parts. Various details on implementing the expert system program developed by the author using CLIPS are given in Appendix B. The following section will demonstrate the application of the proposed expert system based methodology for set-up planning with the help of the illustrative example of rotationally symmetrical part shown in Figure 3.10.

4.2 Illustrative Example and Results

In this section, two illustrative examples are used to demonstrate the application of the proposed methodology for set-up planning.

4.2.1 Illustrative Example 1

The set-up plan for machining the part shown in Figure 3.10(a) has to be determined. The raw stock is a forged blank made of 4140 alloy steel. The part contains the following 30 machining features: features 1, 3, 14 of the type face, features 2, 4, 6, 7, 8, 9, 13 of the type external step, feature 5 of the type external taper, features 10, 12 of the type groove, feature 11 of the type external thread and the features 15, 16, 17, 18 of the type hole, feature 19 (8 in number) of the type hole and feature 20 (4 in number) of the type slot. Apart from features 19 and 20, all other features are rotationally symmetrical. The TAD for machining 1, 2, 3, 15 and 16 is from the left, and the TAD for machining 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 18 and 20 is from the right, and the possible TAD for machining 4, 17 and 19 may be either from the left or from the right. The feature 4 has geometric tolerance relationships of 0.1μm, 0.3μm and 0.2μm respectively with the features, 2, 7 and 13. The machining operations selected to produce each feature of the part that have been obtained from the process selection module developed in Chapter 3 are shown in Table 4.1. The machine tool selected for machining the features 1 to 18 is CNC lathe and that for machining the features 19 and 20 is CNC milling machine. The above information, which constitutes the input to the expert system based set-up planner, is represented in the input data format of CLIPS following the syntax given in the template definition of "features" and "operations", and stored in data files with the extension .clp. The data files are then loaded into the CLIPS environment and the expert system program is executed. Table 4.2 summarises the results of the output generated by the expert system that includes the group of operations in each set-up, the sequence of operations and the method of locating and clamping the part in each set-up. It took about 2 minutes on a Pentium 4, 1.7 GHz PC with 1GB RAM to generate the above output. The set-up plan for machining the rotationally symmetrical features on CNC lathe has been generated automatically by the expert system, while that for machining the features 19 and 20 has been selected manually. The results in Table 4.2 indicate that the machining of the rotationally symmetrical features of the part on CNC lathe has to be carried out in two set-ups. The machining operations on 1, 2, 3, 15 and 16 have been assigned to the left set-up since the TAD for machining the above features is from the left. Similarly the machining operations on 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 and 18 have been assigned to the right set-up since the TAD for machining the above features is from the right. The machining operations on 4 have been assigned to the left set-up. This is so because feature 4, which has geometric tolerance relationship with features 2, 7 and 13, has a tighter geometric tolerance relationship with the feature 2 as compared to that with the features 7 and 13, and so the machining operation on 4 has been grouped along with that on 2 in the same set-up, namely left. The machining of features 19 and 20 has to be carried out in two different set-ups on the CNC milling machine as shown in Table 4.2. Also in Table 4.2, the different machining operations have been listed in the sequence in which they are to be actually performed in each set-up. The expert system has taken into consideration the various precedence constraints as well as manufacturing logic to be followed in sequencing the operations to come up with the above operations sequence. Further, the features to be used for locating and clamping the part for each set-up have been identified in Table 4.2. The expert system has taken into consideration the heuristic principles discussed in section 4.1.2.3 to select the datum features. For carrying out machining operations of the left set-up on CNC lathe, the cylindrical surface of the external step feature 13 has been selected for clamping the part and the end face 14 for locating it. For carrying out the machining operations of the right set-up on CNC lathe, the cylindrical surface of the external step feature 4 has been selected for clamping the part and the vertical face 3 for locating it. For carrying out machining operations on feature 19 on CNC milling machine, the cylindrical surface of the external step feature 13 has to be used for clamping the part and the end face 14 for locating it. For carrying out machining operations on feature 20, the cylindrical surface of the external step feature 4 has to be used for clamping the part and the vertical face 3 for locating it.

Table 4.1. Machining Operation Sequences for Producing Different Features of the Part Shown in Figure 3.10(a)

Feature identifier	Feature type	Operation description	Operation identifier
1	Face	Rough turn	101
		Semi finish turn	102
		Finish turn	103
2	External step	Rough Turn	201
		Semi finish Turn	202
		Finish Turn	203
3	Face	Rough turn	301
		Semi finish turn	302
		Finish turn	303
4	External step	Rough Turn	401
		Semi finish Turn	402
		Finish Turn	403
5	External taper	Rough Turn	501
		Semi finish Turn	502
		Finish Turn	503
6	External step	Rough Turn	601
		Semi finish Turn	602
		Finish Turn	603
7	External step	Rough Turn	701
		Semi finish Turn	702
		Finish Turn	703
8	External step	Rough Turn	801
		Semi finish Turn	802
		Finish Turn	803

Table 4.1. (Continued) Machining Operation Sequences for Producing Different Features of the Part Shown in Figure 3.10(a)

Feature identifier	Feature type	Operation description	Operation identifier
9	External step	Rough Turn	901
		Semi finish Turn	902
		Finish Turn	903
10	Groove	Groove turning (two passes)	10
11	External thread	Rough Turn	1101
		Semi finish Turn	1102
		Finish Turn	1103
		Threading	11
12	Groove	Groove turning (two passes)	12
13	External step	Rough Turn	1301
		Semi finish Turn	1302
		Finish Turn	1303
14	Face	Rough turn	1401
		Semi finish turn	1402
		Finish turn	1403
15	Hole	Drill	1501
		Rough Bore	15001
		Semi finish Bore	15002
		Finish Bore	15003
16	Hole	Drill	1601
		Rough Bore	16001
		Semi finish Bore	16002
		Finish Bore	16003

Table 4.1. (Continued) Machining Operation Sequences for Producing Different Features of the Part Shown in Figure 3.10(a)

Feature identifier	Feature type	Operation description	Operation identifier
17	Hole	Deep hole drill	17
18	Hole	Drill	1801
		Rough Bore	18001
		Semi finish Bore	18002
		Finish Bore	18003
19	Hole	Drill	1901
		Rough Bore	19001
		Semi finish Bore	19002
		Finish Bore	19003
20	Slot	Rough mill	2001
		Semi finish mill	2002

Table 4.2. Set-up Plan Recommended by the Expert System Based Set-up Planner for the Part Shown in Figure 3.10(a)

Machine tool	Set-up	Sequential order of machining operations	Datum features	
			Clamping	Locating
CNC lathe	Left	101 201 401 301 102 202 402 302 103 203 403 303 17 1601 16001 1501 15001 16002 15002 16003 15003	13	14
	Right	1401 1301 11101 901 701 601 801 501 1402 1302 11102 902 702 602 802 502 1403 1303 11103 903 703 603 803 503 12 10 11 1801 18001 18002 18003	4	3
CNC milling machine	-	1901 19001 19002 19003	13	14
CNC milling machine	-	2001 2002	4	3

4.2.2 Illustrative Example 2

This section illustrates how the set-up plan for machining the part shown in Figure 3.10(b) has been determined by using the proposed expert system based methodology. The part contains the following 15 machining features: features 1, 8 of the type face, features 2, 4, 5 of the type external step, feature 3 of the type external taper, feature 6 of the type groove, feature 7 of the type external thread and the features 9, 10, 11, 12 and 13 of the type hole, feature 14 (2 in number) of the type keyway and feature 15 (8 in number) of the type hole. All the features except 14 and 15 are rotationally symmetrical. It may be also noted that the features, namely the thread feature number 7 and the keyway feature number 14 are overlapping. The Tool Access Direction (TAD) for machining 1, 9, 10, 11 and 12 is from the left, and the TAD for machining 3, 4, 5, 6, 7, 8, and 14 is from the right, and the possible TAD for machining 2, 13 and 15 may be either from the left or from the right. The feature 13 has geometric tolerance relationships of 0.1μm and 0.2μm respectively with the features, 5 and 9. The machining operations selected to produce each feature of the part that have been obtained from the process selection module are shown in Table 4.3. The machine tool selected for machining the features 1 to 13 is CNC lathe and that for machining the features 14 and 15 is CNC milling machine. All the above information constitutes the input to the expert system based set-up planner and is represented in the input data format of CLIPS following the syntax given in the template definition of "features" and "operations", and stored in data files with the extension .clp. The data files are then loaded into the CLIPS environment and the expert system program is executed. Table 4.4 summarises the results of the output generated by the expert system that includes the group of operations in each set-up, the sequence of operations and the method of locating and clamping the part in each set-up. It took about 2 minutes on a Pentium 4, 1.7 GHz PC with 1GB RAM to generate the above output. The set-up plan for machining the rotationally symmetrical features on CNC lathe has been generated automatically by the expert system, while those for machining the features 14 and 15 have been selected manually. The results of the output generated by the expert system which have been summarized in Table 4.4 indicate that the machining of the rotationally symmetrical features of the part on CNC lathe has to be carried out in two set-ups. The machining operations on 1, 9, 10, 11 and 12 have been assigned to the left set-up since the TAD for machining the above features is from the left. Similarly the machining operations on 3, 4, 5, 6, 7 and 8 have been assigned to the right set-up since the TAD for machining the above features is from the right. The machining operations on 2 have been arbitrarily assigned to the left set-up since the TAD for machining it may be either from the left or from the right. The machining operation on 13 has been assigned to the right set-up. This is so because feature 13, which has geometric tolerance relationship with features 5 and 9, has a tighter geometric tolerance relationship with the feature 5 as compared to that with the feature 9, and so the machining operation on 13 has been grouped along with that on 5 in the same set-up, namely right. The machining of features 14 and 15 has to be carried out in two different set-ups on the CNC milling machine as shown in Table 4.4. Also in Table 4.4, the different machining operations have been listed in the sequences in which they are to be actually performed in each set-up. The expert system has taken into consideration the various precedence constraints as well as manufacturing logic to be followed in sequencing the operations to come up with the operations sequences. It is to be noted that the sequence of machining operations on overlapping features, namely 7 and 14 as recommended by the expert system is machining of 7 followed by machining of 14. This is in accordance with good manufacturing practice. Further, the features to be used for locating and clamping the part for each set-up have been identified in Table 4. The expert system has taken into consideration various heuristic principles to select the datum features. For carrying out machining operations of the left set-up on CNC lathe, the cylindrical surface of the external step feature 4 has been selected for clamping the part and the right edge of 4 for locating it. For carrying out the machining operations of the right set-up on CNC lathe, the cylindrical surface of the external step feature 2 has been selected for clamping the part and the vertical face 1 for locating it. For carrying out machining operations on feature 15 on CNC milling machine, the cylindrical surface of the external step feature 4 has to be used for clamping the part and the right edge of 4 for locating it. For carrying out machining operations on feature 14, the cylindrical surface of the external step feature 2 has to be used for clamping the part and the vertical face 1 for locating it.

Table 4.3. Machining Operation Sequences for Producing Different Features of the Part Shown in Figure 3.10(b)

Feature identifier	Feature type	Operation description	Operation identifier
1	Face	Rough turn	101
		Semi finish turn	102
		Finish turn	103
2	External step	Rough Turn	201
		Semi finish Turn	202
		Finish Turn	203
3	External taper	Rough Turn	301
		Semi finish Turn	302
		Finish Turn	303
4	External step	Rough Turn	401
		Semi finish Turn	402
		Finish Turn	403
5	External step	Rough Turn	501
		Semi finish Turn	502
		Finish Turn	503
6	Groove	Groove turning (two passes)	6
7	External thread	Rough Turn	7001
		Semi finish Turn	7002
		Finish Turn	7003
		Threading	7

Table 4.3. (Continued) Machining Operation Sequences for Producing Different Features of the Part Shown in Figure 3.10(b)

Feature identifier	Feature type	Operation description	Operation identifier
8	Face	Rough turn	801
		Semi finish turn	802
		Finish turn	803
9	Hole	Drill	901
		Rough Bore	9001
		Semi finish Bore	9002
		Finish Bore	9003
10	Hole	Drill	1001
		Rough Bore	10001
		Semi finish Bore	10002
		Finish Bore	10003
11	Hole	Drill	1101
		Rough Bore	11001
		Semi finish Bore	11002
		Finish Bore	11003
12	Hole	Drill	1201
		Rough Bore	12001
		Semi finish Bore	12002
		Finish Bore	12003
13	Hole	Deep Hole Drill	1301
14	Keyway	Rough mill	1401
		Semi finish mill	1402
15	Hole	Drill	1501
		Counter Bore	1502

Table 4.4. Set-up Plan Recommended by the Expert System Based Set-up Planner for the Part Shown in Figure 3.10(b)

Machine tool	Set-up	Sequential order of machining operations	Datum features	
			Clamping	Locating
CNC lathe	Left	101 201 102 202 103 203 1101 11001 901 9001 1201 12001 1001 10001 11002 9002 12002 10002 11003 9003 12003 10003	4	4
	Right	801 7001 501 401 301 802 7002 502 402 302 803 7003 503 403 303 6 7 1301	2	1
CNC milling machine	-	1401 1402	2	1
CNC milling machine	-	1501 1502	4	4

4.3 Discussions

In this section, the differences between the expert system systems based set-up planning approach, developed by the author, and other approaches developed by previous researchers will be discussed along with some of the important advantages that stand to be gained from the proposed approach.

4.3.1 Comparison of the Proposed Expert System Approach with Other Approaches Developed by Previous Researchers

Three key issues in set-up planning have been addressed in here, namely, the set-up formation, the operation sequence determination and the selection of locating and clamping surfaces. As noted from the review of literature, the problem of set-up formation had been solved by previous researchers using various approaches such as algorithms and graphs, mathematical programming, expert systems and unsupervised neural networks; the problem of determining the operation sequence had been solved using various approaches such as constraint programming, mathematical programming, expert systems, fuzzy logic and Hopfield neural networks; the datum selection problem had been solved using various approaches such as algorithms and graphs, expert systems and back-propagation neural networks. In the present work, all of the above three set-up planning problems have been solved using the expert system based approach.

Further, most of the expert systems based set-up planning approaches, that were proposed by previous researchers, had been developed for applications in the prismatic parts domain. The author, in the present work, has presented an expert system based set-up planning methodology for application in the domain of the rotationally symmetrical parts. It is to be noted that although the nature of the set-up planning problem in case of the above two application domains appear to be fundamentally similar, however, a different approach for implementing set-up planning needs to be adopted for each of them. This is so because in case of machining rotational parts, only two set-ups are possible as each feature can have only 2 possible TADs, while in the case of prismatic parts, more than two set-ups are possible as each machining feature may have upto 6 possible TADs.

Furthermore, in most of the previously reported expert systems based set-up planning approaches, a mixture of the expert systems and some algorithmic approach had been adopted. One significant limitation of such an approach is that the resulting system tends to be inflexible, responding poorly to new situations; particularly, when it comes to modification of the existing algorithms that might be necessary in order to bring about any changes in the current set-up planning knowledge or acquire new knowledge, it might require rewriting of the original program for the algorithm, which could turn out to be a tedious and time-consuming exercise in certain situations. In the present work, the author has adopted a pure expert systems approach to solve the different problems in set-up planning. The factual data or the input description of the part being planned, the heuristic rules representing the domain knowledge on set-up planning and the general problem solving knowledge to control the way in which the rules are to be applied to the facts have been organised into different modules of the expert system, such as the database, the knowledge base and the inference engine. The modular nature of the expert systems and separation of control knowledge or inference engine from the knowledge base gives added flexibility to the proposed approach. Any modification of the current set-up planning knowledge, if necessary, can be done by simply modifying the rules in the knowledge base of the expert system that is less time consuming than having to modify the original program as in the case of approaches that use algorithms. Also new knowledge can be easily acquired by the expert systems through introduction of new rules to its knowledge base as will be explained below. The expert system based method adopted by the author also offers a generative approach to CAPP that can help to overcome several of limitations of the variant approach. Unlike the variant approach, the method presented is not just limited to similar parts previously planned. Moreover it is able to generate set-up plans automatically and consistently over and over again. By eliminating the need for expert process planners while generating the set-up plan, the present method can help ensure that the process plan is free from sources of human error that may otherwise occur with the variant CAPP approach. Another advantage of the developed expert system based methodology is with regard to the computation time to generate the set-up plans. The developed expert system was tested on a variety of example parts. The results of the tests show that the computation time for generating the set-up plan by the developed expert system is reasonably fast so that it can be used to quickly solve most practical problems. This, in turn, translates directly to the planning time saved in the process planning stage and hence reduces the cost.

4.3.2 Flexibility and Adaptability of the Developed Expert System for Set-Up Planning

The illustrative examples have demonstrated the potential for application of the proposed expert system based methodology for set-up planning of a wide variety of rotationally symmetrical parts. The module for set-up planning developed as part of this research work can handle situations involving different types of the machining features that are commonly encountered in rotationally symmetrical parts such as internal surfaces of revolution like holes, external surfaces of revolution like external diameters, external taper, external thread, groove, face, and other features such as slot, keyway, and so on. In addition it can also handle cases where parts may contain overlapping features, e.g. an external thread and an overlapping keyway. The developed expert system has a flexible architecture that enables it to adapt readily to new, different situations through modifications of its knowledge base that will be explained in the following paragraphs.

The modular nature of the proposed expert system and the separation of control knowledge (or inference engine) from the databases containing the procedural knowledge or domain knowledge and the declarative knowledge about the part being planned give flexibility in terms of making changes to the process planning system quickly and expanding its capability to accommodate new, different situations. For example, in future, situations may arise where, due to major changes in the product design, a completely new set of machining features may be encountered in the part, for which set-up planning rules do not exist in the knowledge base of the expert system. Also there may be a situation where a new strategy for more optimal method of generating set-up plans is found that calls for some changes in the existing set-up planning rules or introduction of a new set of rules. In such situations, it will suffice to merely modify the knowledge base of the proposed expert system. The following explains how it can be accomplished.

It may be recalled that in the developed expert system, a feature template such as the ones shown in Figure 4.9 is used to describe the format for representation of the input data about features. It has one slot for each of the following namely, the feature identifier, name of the feature (e.g. external step, hole, etc.), type of feature (internal or external), sub-type of feature (primary or secondary), feature attributes such as diameter, and so on. In the case where new machining features are encountered for which set-up planning rules do not exist in the knowledge base of the expert system, first all the slots of the feature template must be updated by adding the new feature type to the existing list, along with any feature attribute that may be necessary for developing the set-up plan. Suppose that the set-up planning system did not initially consider the feature, "keyway" in the list of machining features that it is designed to handle. The feature template before inclusion of the keyway feature is shown in Figure 4.9(a). In order to include the keyway feature, first of all, the "allowed-symbols" list of the "name" slot in feature template definition has to be expanded by including the keyword "keyway" (shown in italics) as in Figure 4.9(b).

```
(deftemplate MAIN::feature
(slot number (type INTEGER) (default ?NONE))
(slot name (type SYMBOL) (allowed-symbols EXTERNAL_STEP FACE GROOVE
HOLE EXTERNAL_TAPER THREAD HOLE))
(slot type (type SYMBOL) (allowed-symbols EXTERNAL INTERNAL))
(slot subtype (type SYMBOL) (allowed-symbols PRIMARY SECONDARY))
.....
..... )
```

(a)

```
(deftemplate MAIN::feature
(slot number (type INTEGER) (default ?NONE))
(slot name (type SYMBOL) (allowed-symbols EXTERNAL_STEP FACE GROOVE
HOLE EXTERNAL_TAPER THREAD HOLE KEYWAY))
(slot type (type SYMBOL) (allowed-symbols EXTERNAL INTERNAL))
(slot subtype (type SYMBOL) (allowed-symbols PRIMARY SECONDARY))
.....
..... )
```

(b)

Figure 4.9 Feature template definitions (a) before and (b) after inclusion of keyway

Next, the rules for carrying out set-up planning for the new machining feature, which in the present case is keyway, will have to be added to the set of existing rules in the knowledge base of the expert system. Also conflicts, if any, which may arise with the existing rules, must be resolved. The following explains how it is accomplished with the aid of some sample rules involving the keyway feature for determination of feature precedence constraint and for operation sequencing.

The sample rule given in Figure 4.10 is used for generating the precedence constraint for machining between the keyway and the other machining features present in the part. The sample_rule states that if there exists a feature "a" that is of the type keyway and if it is secondary to the feature "b", then the precedence for machining the features a and b will be machining of b followed by machining of a. Thus if there exists a fact about a thread feature in the facts list, and another fact about a keyway feature that is secondary to the thread feature, then the above rule generates a precedence constraint that machining of the thread feature has to be done prior to that of the keyway.

```
(defrule sample_rule
(feature (number ?a) (name KEYWAY) (type EXTERNAL) (secondary_feature_to
$? ?b $?))
=>
(assert (feature_precedence ?b ?a)))
```

Figure 4.10 Sample rule for determining the precedence constraint between the keyway and the other features

The sample_rule1 given in Figure 4.11 is used for operation sequencing if a machining operation of an external step feature is encountered and suppose that the sample_rule1 existed before in the knowledge base of the expert system before inclusion of the keyway feature. Also suppose that sample_rule2 did not exist before; it has been newly created so as to carry out operation sequencing taking into account the newly included keyway feature. Now if the antecedent (IF) parts of both the above rules happen to be satisfied by the facts in the database, the two rules will constitute what is known as a conflict set. It is important to resolve the conflicts of the newly created rules with any pre-existing rules. The following explains how it is accomplished. Notice that the salience value or priority in execution of the sample_rule1 is 99 and the salience value for the newly entered sample_rule2 has been set at 9, which is lower than that for sample_rule1. This will cause the sample_rule1 to be executed prior to that of the sample_rule2, if the antecedent (IF) parts of both the rules happen to be satisfied simultaneously. It, in turn, signifies that the machining operation of the external step will be assigned to the operation sequence ahead of the machining operation of the keyway. This is in accordance with the manufacturing logic that the machining of a primary feature, which in this case is the external step, should precede that of a secondary feature, which in this case is keyway. The resolution of conflict with a pre-existing rule in the present case has been accomplished by rule prioritization. It is important to resolve the conflicts of the newly entered rules with the pre-existing rules, which have been shown to be accomplished in the above example through rule prioritization.

```
(defrule sample_rule1
        (declare (salience 99))
        ?f1 <- (opn (number ?n1) (machining_stage rough) (setup-cluster left)
(preceding_opn ?n2))
        (operation (number ?n1) (on-feature ?N1))
        (feature (number ?N1) (name EXTERNAL_STEP))
        (test (not (= ?n2 0)))
        (opn (number ?n2) (machining_stage rough) (setup-cluster left))
=>      (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
        (if (subsetp (create$ ?n2) (create$ ?*sequence-left-cluster*))
          then
          (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-
cluster*))))

(defrule MAIN::sample_rule2
        (declare (salience 9))
        ?f1 <- (opn (number ?n1) (machining_stage rough) (setup-cluster left)
(preceding_opn ?n2))
        (operation (number ?n1) (on-feature ?N1))
        (feature (number ?N1) (name KEYWAY))
        (test (not (= ?n2 0)))
        (opn (number ?n2) (setup-cluster left))
=>      (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
        (if (subsetp (create$ ?n2) (create$ ?*sequence-left-cluster*))
          then
          (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-
cluster*))))
```

Figure 4.11 Sample rules for operation sequencing involving the keyway

Finally the performance of expert system for set-up planning system must be verified by presenting situations involving parts containing the new feature types to see if it can handle the new situations. In a similar manner, modification of existing rules or introduction of new rules may be accomplished, if a new more optimal method of generating the set-up plans is found. The method described above for modification of the knowledge base in order to accommodate the new situations is easier to accomplish owing to the flexible architecture of the expert system, than having to modify the original program as in the case of approaches using algorithms and graphs, which can turn out to be quite tedious and time consuming. This enables the expert system to respond more quickly to new situations.

4.3.3 Integration of the expert system module for set-up planning with other CAPP submodules

Although in the present work, the expert system for set-up planning has been developed as a stand alone application program, it is possible to extract the source code of the program in C and then embed it within any user defined C application program. This feature enables the expert system module for set-up planning to be integrated with other modules of the CAPP system such as modules for machining operation selection, cutting tool and machine tool selection, and module for automatic feature extraction from CAD system. From the above discussions, it is clear that the expert system based set-up planning approach developed by the author has lot of potential to be used as an alternative to the traditional approaches and those developed by previous researchers.

CHAPTER 5

CONCLUSIONS

The previous chapters have presented the background, the methodologies developed as part of this research work and the results of the research in details. The present chapter concludes the thesis by summarising the research and the important conclusions, identifying the main research contributions and by exploring scope for further work and future research directions.

5.1 Summary and Comments

The research work reported in the thesis has explored applications of Artificial Intelligence techniques for automating two important decision making tasks in generative Computer-Aided Process Planning (CAPP) systems namely,

- selection of machining operations and
- set-up planning in rotationally symmetrical parts.

5.1.1 Summary and Comments on the Work Accomplished on Selection of Machining Operations in Rotationally Symmetrical Parts

A back-propagation learning based neural networks methodology has been developed for automating the task of selection of machining operation sequences. It takes in as input the attributes of each feature and is capable of automatically selecting all the possible alternative sequences of machining operations to produce that feature. The commercial PC-based software package *NeuFrame Version* 4 (2000) has been used to simulate the neural network operation. The types of the features that have been considered are internal surfaces of revolution like holes, external surfaces of revolution like external diameters, external taper, external thread, groove, face, and other features such as slot, keyway, which are commonly encountered in rotationally symmetrical parts. The detailed description of development of the neural network methodology for machining operations selection has been given in Chapter 3. The key issues discussed in this regard include the gathering of domain knowledge for formulating the thumb rules, the topology of the neural network, the format of representation of input and output decision variables, the training and validation of the neural network and how to accomplish modification of the knowledge base of the neural network so as to adapt it to new situations. The potential for application of the developed neural network methodology has been illustrated with the help of examples of rotationally symmetrical parts. The following are some of the important comments and conclusions based on the research carried out by the author regarding development of the neural network methodology for machining operations selection:

a) From the literature review, a limitation observed with the previously developed neural network based approaches by Knapp et al (1992) and Devireddy et al (1999, 2002) is that there are no guidelines for choosing the input patterns of training examples, which is often difficult, and a tedious and time-consuming exercise of trial and error. Also an issue that has not been adequately addressed by previous researchers is that whether any prior domain knowledge on machining operations selection could be taken advantage of. The prior domain knowledge is well known to reduce the complexity of learning in neural network. Further most of the previously developed neural network models tend to recommend a single machining operation sequence for a given feature of the part. But, in practice, the process planning system needs to know the alternative ways of machining a part for use in developing process planning alternatives. Keeping the above in mind, the author has developed a neural network based methodology for selection of all the possible alternative operation sequences for machining a given feature of the part by prestructuring it with prior knowledge and that can help to ease the aforementioned problems of choosing training examples.

b) A back-propagation learning based feed forward neural network architecture has been adopted by the author. The neural network has been pre-structured with prior knowledge on the problem domain i.e. machining operations selection in the form of thumb rules. It has been achieved by developing two forms of representation for the input data to the neural network. The external representation is used to enter the crisp values of the input decision variables (namely the type of feature and the various attributes namely, diameter or width, tolerance and surface finish) to the neural network. The purpose of internal representation is to categorize the crisp values of various feature attributes that are entered into certain sets; these sets correspond to all the possible different ranges of the feature attributes encountered in the antecedent 'IF' part of the thumb rules for machining operation selection. The translation of the input data from the external to the internal representation format is performed automatically with the aid of simple classification rules developed by the author.

c) Since the developed network model has been prestructured with prior domain knowledge in the form of thumb rules, the quality of the domain knowledge that is used for formulating the thumb rules strongly influences the quality of the neural network model. So the acquisition of domain knowledge has to be performed carefully. The domain knowledge for the present work was collated primarily from various sources such as handbooks and textbooks on machining and based on interviews with the experts in the field of process planning as well as discussion with skilled machinists. In practice for a given company, it is necessary to also consult the appropriate vendor catalogues of the manufacturing equipments present in the shop floor and company specific manufacturing process handbooks for detailed information about process capabilities of various machining operations.

d) Most of the previously developed neural network models recommend a single machining operation sequence for a given feature of the part with each output layer node of the neural network representing a single operation. The neural network model developed by the author recommends all the possible operation sequences for machining a given feature of the part with each output layer node representing a possible sequence of machining operations. This will help the process planning system in exploring all the possible alternative process plans, and then choose the most appropriate process plan from among them, depending upon the machine tool and cutting tool availability, manufacturing cost, various optimisation criteria and/or any other constraints specific to the shop floor.

e) In the neural network approaches by previous researchers, as pointed out earlier, there are no guidelines of choosing input patterns of the training examples for the neural network. The input variable values for the input patterns are, at first, randomly selected from the entire range and the neural network is trained. Then the generalisation capability of the network is verified by validation tests, failing which the training dataset has to be modified by adding the situations of the validation dataset, which have generated greater errors. In the present approach proposed by the author, a more systematic method of choosing input patterns for the training examples has been illustrated. The thumb rules developed for selection of machining operation sequences are used to serve as guidelines during the preparation of training examples. The input patterns have to be chosen in such a way that they activate one or more of these thumb rules. It is done by selecting the input variable values that fall in the ranges related to those rules rather than randomly selecting the values of input variables from their entire range. This method of choosing input patterns of training examples simplifies to a great extent the process of preparation of training examples and it has been found to help ensuring in a better way that the entire problem domain is represented and requires substantially lesser number of training examples. This helps in increasing the computational efficiency during training, thereby reducing the time taken to develop the overall process planning system and hence making it cost effective and attractive for industrial applications in process planning of complex components.

f) The results of the tests carried out by the author on a variety of example parts show that the computation time for generating machining operation sequences by the neural network model developed by the author is reasonably fast so that it can be used to quickly solve most practical problems. This, in turn, saves the planning time during the process planning stage and hence reduces the cost.

g) The process planning methodology developed by the author offers a generative approach to CAPP that can help to overcome a number of limitations of the variant approach. Unlike the variant approach, the method presented is not just limited to similar parts previously planned. Moreover it is able to generate process plans automatically and consistently over and over again. By eliminating the need for expert process planners while generating the process plan, the present method can also significantly cut down the time and hence the costs in process planning and ensures that the process plan is free from sources of human error that may otherwise occur with the variant CAPP approach. In the event of any major changes in product design or the shop floor manufacturing environment, the method presented may be quickly adapted to accommodate the above changes by necessary modifications of the knowledge base through mere retraining of the network.

h) The illustrative examples have demonstrated the potential for application of the developed neural network based methodology for machining operations of a wide variety of rotationally symmetrical parts. The proposed method overcomes various limitations of other generative CAPP approaches such as decision trees and expert systems by its ability to automatically acquire knowledge by learning from training examples and generalize beyond the set of training examples within the limits set by the training dataset. Further it has a flexible architecture and it is characterized by its ability to adapt itself readily to changes in the manufacturing environment. Both the decision trees and the expert systems are relatively static in terms of representing the process planning knowledge. In the proposed method by the author, modification of the process planning knowledge base can be accomplished relatively more easily and quickly by merely retraining the network. In future, for example, in the light of new developments in the manufacturing technology, if a new process is developed or a more optimal sequence is found, the knowledge base will have to be modified. In order to accomplish this, first the new knowledge will have to be expressed in the form of thumb rules. Then the antecedent and consequent parts of the new rules will have to be scanned and if a new range of one of the input decision variables or operation sequence is found, a new node must be created in the input or output layer of the network. If the new rules are not in direct conflict with the pre-existing rules set in the knowledge base, then it might be sufficient to just add, to the previous training dataset, a new set of training examples so as to activate the new rules and retrain the network. If, however, one or more pre-existing rules becomes irrelevant as a result, they have to be identified and the nodes representing the range of input decision variables and operation sequences corresponding to the affected rules will have to be removed from the input and output layers; also the training examples meant for activating these rules will have to be removed from the previous training dataset and the network will have to be retrained. Another advantage that the present method offers is that, as compared to the decision trees or expert systems, it leads to faster inference, owing to the parallel architecture of the network. From the above discussions, it is evident that the CAPP approach by the author has lot of potential to be used as an alternative to the traditional variant and generative CAPP approaches. By this methodology, workpieces of complex shapes can be handled by investing a very limited amount of time, making it attractive and cost effective for industrial applications.

5.1.2 Summary and Comments on the Work Accomplished on Set-up Planning in Rotationally Symmetrical Parts

An expert systems based methodology has been developed for automating the three main decision making tasks in set-up planning, namely the set-up formation, operation sequencing and selection of locating and clamping surfaces for each set-up in CAPP systems for rotationally symmetrical parts. It takes in as input the data files containing information such as the types of features present in the part, the dimensions, the Tool Access Directions (TADs) in order to machine each feature and the geometric tolerance relationship between features, which need to be extracted from the original part design as well as the operation sequences selected for machining each feature, which need to be obtained from the process selection module and it is capable of generating set-up plans automatically. It has been implemented on a PC by using the CLIPS rule-based expert system shell (Giarratano et al 1998). The proposed expert system based methodology for set-up planning caters to the rotationally symmetrical parts containing different features such as internal surfaces of revolution like holes, external surfaces of revolution like external diameters, external taper, external thread, groove, face, and other features such as slot, keyway, and further parts that may contain overlapping features, e.g. an external thread and an overlapping keyway. The detailed description of development of the expert system methodology for set-up planning has been given in Chapter 4. The key issues discussed in this regard include the development of the overall structure of the expert system, the database, the knowledge base and the inference engine for solving the problems of set-up formation, operation sequencing and selection of locating and clamping surfaces and finally how to accomplish modification of the knowledge base to adapt it to new situations. The examples of some industrially-relevant rotationally symmetrical workpieces have been analysed using the proposed approach to demonstrate its potential for application in the real manufacturing environment. The following are some important conclusions based on the research carried out by the author regarding development of the expert system based methodology for automating the different set-up planning tasks:

a) From the literature review, it was observed that the potential for application of expert systems for set-up planning particularly in rotational parts has yet to be fully explored. Most of the previous research efforts so far have been limited to the prismatic parts domain. Also, in most of the previously reported expert systems based approaches, a mixture of the expert systems and some algorithmic approach had been adopted to solve the set-up planning problems. The limitation of it is that the resulting system tends to be inflexible, responding poorly to new situations. In particular, when it comes to modification of the current set-up planning knowledge or acquiring new knowledge, it might require rewriting of the original program for the algorithm. Keeping the above in mind, the author has developed a purely expert system based methodology for solving the different set-up planning problems in CAPP systems for machined rotationally symmetrical parts.

b) In order to represent the input data for the expert system namely, the different features present in the part and the operations required for machining each feature, a format for representation has been developed by the author using template that is the commonly used input data representation format in CLIPS. There are two ways by which the above input data can be entered into the developed expert system. The input data can be saved as a data file with the extension .clp and loaded from the file into the expert system environment at the time of execution. Alternatively, it may be also directly entered manually by typing through a user interface.

c) A rule based knowledge base has been developed for solving each of the problems of set-up formation, operation sequencing and datum selection. To carry out set-up formation, a set of rules has been developed, which are capable of clustering the machining operations into set-ups taking into consideration the TAD of the corresponding features and the relative tolerance relationships between them. Further, a set of rules has been proposed to establish the various feature precedence constraints and to determine the operation sequence in each set-up, subject to the above precedence constraints as well as manufacturing logic for ordering the operations. Finally a set of rules, based on heuristic principles developed by the author are used for selection of the locating and clamping features in each set-up. The inference engine for the CLIPS expert system shell is based on a forward chaining strategy.

d) The expert system developed for set-up planning was tested on a variety of example problems (parts). The results of the tests show that the computation time for generating the set-up plan by the developed expert system is reasonably fast so that it can be used to quickly solve most practical problems. This, in turn, saves the planning time while performing the process planning tasks and hence reduces the cost.

e) The practicality and usefulness of the expert system strongly depends on the quality, consistency and completeness of the domain knowledge that is used for formulating the rules of the expert system. So the acquisition of domain knowledge needs to be done carefully. The knowledge necessary for formulating the rules in the present work for performing the various set-up planning tasks has been based on the heuristic knowledge and expert knowledge from various sources such as handbooks and textbooks on machining, and based on interviews with the experts in the field of process planning as well as discussion with skilled machinists and direct observations of actual machining of workpieces in shop floor.

f) The modular nature of the expert systems and separation of control knowledge or inference engine from the knowledge base gives added flexibility to the proposed approach in terms of making changes to the process planning system quickly and expanding its capability to adapt it to new, different situations. Any modification of the current set-up planning knowledge can be accomplished by simply modifying the rules in the knowledge base of the expert system that is less time consuming than having to modify the original program as in the case of approaches that use algorithms. Also new knowledge can be easily acquired by the expert systems through introduction of new rules to its knowledge base. However, while updating the knowledge base, care must be exercised to ensure that the new rules are consistent with the existing rules. If there is any contradiction of the newly entered rules with one or more existing rules, it must be accounted for while updating the knowledge base. By this methodology, the set-up planning of rotationally symmetrical machined parts of complex shapes can be accomplished automatically by investing a very limited amount of time, making it attractive, cost effective and practical for use in industrial applications.

5.2 Contributions

The research work undertaken by the author makes a number of contributions in the area of applicability of the Artificial Intelligence techniques such as the Artificial Neural Networks and Expert Systems in solving the various process planning problems in generative CAPP systems for machined rotationally symmetrical parts. The following summarizes the major contributions:

a) An intelligent back-propagation neural network based methodology has been developed for automating the selection of operations for machining a feature of the part (given its dimensions, tolerance and surface finish specifications) in CAPP systems for rotationally symmetrical parts. The implementation of the developed neural network based methodology has been illustrated with some examples.

b) The proposed neural network model takes advantage of the prior domain knowledge (in the form of heuristic or thumb rules) on machining operations selection, which in turn helps to reduce the complexity of training. It has been achieved by developing a format for representation of the input decision variables for machining operations selection to the neural network, by prestructuring the input layer with prior domain knowledge.

c) The neural network model developed by the author recommends all possible operation sequences for machining a given feature of the part with each output layer node representing a possible sequence of machining operations. This enables the process planning system in exploring all possible alternative process plans before choosing the most appropriate plan from among them.

d) A systematic method of choosing the input patterns of the training examples for the neural network has been illustrated in this work. It simplifies the process of preparation of training examples to a great extent and can help ensuring in a better way that the entire problem domain is represented.

e) The proposed neural network based methodology can be applied for machining operations selection of a wide variety of rotationally symmetrical parts. Further, it has a flexible architecture and it is characterized by its ability to adapt itself readily to changes in the manufacturing environment.

f) An intelligent expert systems based methodology has been developed for solving the different set-up planning problems in CAPP systems for rotationally symmetrical parts using the CLIPS expert system shell. The implementation of the developed expert system based methodology has been illustrated with some examples.

g) A format of representation of the input data to be entered in the expert system for set-up planning has been developed by using template that is the commonly used input data representation format in CLIPS.

h) Further a set of knowledge based rules have been developed for solving each of the different set-up planning problems subject to a set of constraints e.g. clustering of operations into set-ups subject to constraints such as tool access directions for machining each feature and the relative tolerance relationships between them, sequencing of operations subject to constraints due to different feature precedence relationships and manufacturing logic for ordering the various machining operations, and finally selection of locating and clamping datums for each set-up based on heuristic principles. The rules have been encoded in the CLIPS language format to implement the expert system.

i) The proposed expert system based methodology can be applied for set-up planning of a wide variety of rotationally symmetrical parts. Further, it has a flexible architecture that enables it to adapt readily to new, different situations through modifications of its knowledge base.

5.3 Scope for Further Work and Future Research Directions

The scope for further work and future research directions are briefly outlined in following paragraphs.

The scope of application of the developed neural network based methodology for machining operations selection may be expanded further to include more complex parts that may contain other machining features that have not been considered in the present work e.g. internal tapers, internal threads, chamfers, counterbore, countersink, round and angular grooves, etc. and by considering more feature attributes. Also the above neural network based methodology may be adopted for machining operations selection in case of mill-turn parts and prismatic parts as well.

The scope of application of the developed expert system based methodology for set-up planning may be also expanded further by considering various other set-up planning constraints e.g. fixturing constraints that have not been considered in the present work. Although the developed expert system can generate good plans, the optimality of the generated plan is not necessarily guaranteed. So there is scope for further optimization of the set-up plan considering all possible operation sequence alternatives by using AI based optimization algorithms such as genetic algorithm. In the present work, the author has assumed the different types of relative tolerances between features to be of equal importance in order to generate the set-up plans, due to lack of a reliable common measure for different types of relative tolerances. However, recently Huang et al (2003) have reported some work on developing a common measure for different types of tolerances between features based on the concept of normalised tolerance. So a direction for future research could be modification of the set-up planning methodology developed by the author in the present work by considering the normalised values of relative tolerances and seeing if it can lead to more optimal set-up plans.

Further work also needs to be done on integration of the CAPP sub-modules with the CAD system and also on integration among various process planning sub-modules within the CAPP system. In this regard, since the source codes for both the neural network based software module developed by the author for machining operations selection and the expert system based software module developed by the author for set-up planning can be extracted in C/C++, it can be easily embedded within a user defined program for the entire CAPP system. Thus it is possible to link the process planning modules developed in this work with each other as well as with other process planning modules such as, modules for automatic feature extraction from CAD systems, modules for machine tool and cutting tool selection, selection of cutting parameters etc., for creation of an integrated modular CAPP system.

REFERENCES

ALLEN, D. K. (1987). An Introduction to Computer-Aided Process Planning. CIM Review. 7-22.

BALAZINSKI, M., BELLEROSE, M., CZOGALA, E. (1994). Application of Fuzzy Logic Techniques to the Selection of Cutting Parameters in Machining Process, Fuzzy Sets and Systems. 63. 307-317.

BRALLA, J. G. (1986). Handbook of Product Design for Manufacturing: A Practical Guide to Low-Cost Production. New York: McGraw-Hill. 1135p.

CHANG, T. C., WYSK, R. A. (1985). An Introduction to Automated Process Planning Systems. New Jersey: Prentice Hall. 230p.

CHANG, T. C. (1990). Expert Process Planning for Manufacturing. Addison-Wesley Publishing Company. 283p.

CHANG, C. A., ANGKASITH, V. (2001). Using Hopfield neural networks for operational sequencing for prismatic parts on NC machines. Engineering Applications of Artificial Intelligence. 14. 357–368.

CHEN, C. L. P., AND STEVEN LECLAIR, R. (1993). Unsupervised Neural Learning for Setup Generation in Process Planning. Proceedings of International Conference on Artificial Neural Networks in Engineering, Saint Louis, Missouri. 663-668.

CHEN, J., ZHANG YF, NEE AYC. (1998). Setup planning using Hopfield net and simulated annealing. International Journal of Production Research. 36. 981-1000.

DAVIES, B. J., DARBYSHIRE, I. (1984). The Use of Expert System in Process Planning. Annals of CIRP. 33:1. 303-306.

DESCOTTE, Y., LATOMBE, J. C. (1981). GARI: A Problem Solver that Plans to Machine Mechanical Parts. Proceedings of IJCAI-7. 776-772.

DEVIREDDY, C. R, GHOSH, K. (1999). Feature-based modeling and neural networks-based CAPP for integrated manufacturing. International Journal Computer Integrated Manufacturing. 12:1. 61-74.

DEVIREDDY, C. R., EID, T., GHOSH, K. (2002). Computer-Aided Process Planning for Rotational Components Using Artificial Neural Networks. International Journal of Agile Manufacturing. 5:1. 27-49.

EVERSHEIM, P. J., HOLZ, B., ZONS, K. H. (1980). Application of Automatic Process Planning and NC Programming. Proceedings of AUTOFACT WEST, Society of Manufacturing Engineers, California. 779-800.

GIARRATANO, J. C., RILEY, G. (1998). Expert systems: principles and programming. Boston: PWS Publishers Co. 597p.

HALEVI, G., WEILL, R. D. (1995). Principles of Process Planning: A logical approach. England: Chapman & Hall. 399p.

HASHMI K, EL BARADIE MA, RYAN M. (1998). Fuzzy Logic Based Intelligent Selection of Machining parameters. Computers Industrial Engineering. 35. 571-574.

HUANG, S. H., ZHANG, H. (1996). Tolerance analysis in setup planning for rotational parts. Journal of Manufacturing Systems. 15. 340-350.

HUANG, S. H., ZHANG HONG-CHAO, SUN SHAN, LI HUA HARRY. (1996). Function Approximation and Neural-Fuzzy Approach to Machining Process Selection. IEEE Transactions on Components, Packaging, and Manufacturing Technology, Part C. 19: 1. 9-18.

HUANG, S. H. (1998). Automated setup planning for lathe machining. Journal of Manufacturing Systems. 17. 196-208.

HUANG, S. H., LIU, Q. (2003). Rigorous Application of Tolerance Analysis in Setup Planning. International Journal of Advanced Manufacturing Technology. 3. 196-207.

JIANG, B., LAU, H., CHANG, F. T. S., JIANG, H. (1999). An automatic process planning system for the quick generation of manufacturing process plans directly from CAD drawings. Journal of Materials Processing Technology. 87. 97-106.

JOSHI, S., VISSA, N. N., CHANG, T. (1988). Expert process planning system with solid model interface. International Journal of Production Research. 26. 863-885.

KHOSHNEVIS B., TAN W. (1995). Automated process planning for hole-making. American Society of Mechanical Engineers, Manufacturing Review. 8:2. 106-113.

KIM, I., SUH, H. (1998). Optimal operation grouping and sequencing technique for multistage machining systems. International Journal of Production Research. 36. 2061-2081.

KING, J. R. (1979). Machine Component Group Formation in Group Technology. Proceedings of Fifth International Conference on Production Research, Amsterdam.

KING, J. R., NAKORNCHAI, V. (1982). Machine Component Group Formation in Group Technology: Review and Extension. International Journal of Production Research. 20:2. 117-133.

KNAPP, GERALD M., WANG, H. (1992). Neural networks in acquisition of manufacturing knowledge. Intelligent Design & Manufacturing. Edited by Andrew Kusiak. New York: John Wiley & Sons Inc.

LEE, D., KIRITSIS, D., XIROUCHAKIS, P. (2001). Branch and fathoming algorithms for operation sequencing in process planning. International Journal of Production Research. 39. 1649-1669.

LI, J., HAN, C., HAM, I. (1987). CORE-CAPP A Company Oriented Semi-Generative Computer Automated Process Planning System. Proceedings of 19th CIRP International Seminar on Manufacturing Systems, Pennsylvania. 219-225.

LINK, C. H. (1976). CAPP-CAM-I Automated Process Planning System. Proceedings of the 13th Numerical Control Society Meeting & Technical Conference.

MAIYO BERNARD S., XIANKUI WANG, CHENGYING LIU. (1999). An Integrated Application of Neural Network, Fuzzy Logic and Expert Systems for Machining Operation Sequencing, Tsinghua Science and Technology. 4: 4.

MARRI, H. B., GUNASEKARAN, A., GRIEVE, R. J. (1998). Computer-Aided Process Planning: A State of Art. International Journal of Advanced Manufacturing Technology. 14. 261-268.

MEI, J., ZHANG, H. C., OLDHAM, W. J. B. (1995). A neural network approach for datum selection. Computers in Industry. 27. 53-64.

MEZIANE, F., VADERA, S., KOBBACY, K., PROUDLOVE, N. (2000). Intelligent systems in manufacturing: current developments and future prospects, Integrated Manufacturing Systems. 11:4. 218-238.

MING, X. G. AND MAK, K. L. (2000). Intelligent setup planning in manufacturing by the neural networks based approach. Journal of Intelligent Manufacturing. 11. 311-331.

MONOSTORI, L., BARASCHDORFF, D. (1992). Artificial Neural Networks in Intelligent Manufacturing. Robotics & Computer Integrated Manufacturing. 9:6. 421-437.

MONOSTORI, L., MARKUS, A., VAN BRUSSEL, H., WESTKAMPER, E. (1996). Machine Learning Approaches to Manufacturing. Annals of CIRP. 45:2. 675-711.

NAU, D. S., CHANG, T. C. (1985). A Knowledge based approach to generative process Planning. Proceedings of Winter Annual Meeting of ASME, PED-21, Florida. 65-72.

NEUFRAME VERSION 4. (2000). Getting Started Manual. Southampton, UK : Neusciences Intelligent Solutions.

OIR. (1983). MULTIPLAN. Waltham, Mass: Organization for Industrial Research, Inc.

ONG, S. K., NEE, A. Y. C. (1997). Automating set-up planning in machining operations. Journal of Materials Processing Technology. 63. 151-156.

OPTIZ, H. (1970). A Classification System to Describe Workpieces. Elmsford, New York: Pergamon.

RADWAN, A. (2000). A practical approach to a process planning expert system for manufacturing processes. Journal of Intelligent Manufacturing. 11. 75-84.

SABOURIN, L., VILLENEUVE, F. (1996). OMEGA, an expert CAPP system. Advances in Engineering Software. 25. 51-59.

SANTOCHI M, DINI G. (1996). Use of neural networks in automated selection of technological parameters of cutting tools. Computer Integrated Manufacturing Systems. 9:3. 137-148.

SCHAFFER, G. (1980). GT via Automated Process Planning. American Machinist. 119-22.

SHUNMUGAM, M. S., MAHESH, P., REDDY, S. V. B. (2002). A method of preliminary planning for rotational components with C-axis features using genetic algorithm. Computers in Industry. 48. 199-217.

TEMPELHOF, K.H. (1979). A System of Computer-Aided Process Planning for Machined Parts. Advanced Manufacturing Technology (Proceedings of 4th IFIP/IFAC Conference, PROLOMAT 1980). Edited by P. Blake. North-Holland, New York: Elsevier.

WANG, H. P., WYSK, R. A. (1986). Applications of microcomputers in automated process planning, Journal of Manufacturing Systems. 5:2. 103-111.

WANG, H., WYSK, R. A. (1988). A knowledge-based approach for automated process planning. International Journal of Production Research. 26: 6. 999-1014.

WANG, H. AND LI, J. (1991). Computer-Aided Process Planning. Amsterdam, Netherlands: Elsevier Science Publishers B V. 336p.

WANG, K. (1998). An integrated intelligent process planning system (IIPPS) for machining. Journal of Intelligent Manufacturing. 9. 503-514.

WONG, T. N., SIU, S. L. (1995). A knowledge-based approach to automated machining process selection and sequencing. International Journal of Production Research. 33:12. 3465-3484.

ZHANG, H., ALTING, L. (1994). Computerized Manufacturing Process Planning Systems. London, UK: Chapman and Hall. 336p.

ZHANG, H., LIN, E. (1999). A hybrid-graph approach for automated setup planning in CAPP. Robotics and Computer Integrated Manufacturing. 15. 89-100.

APPENDIX A

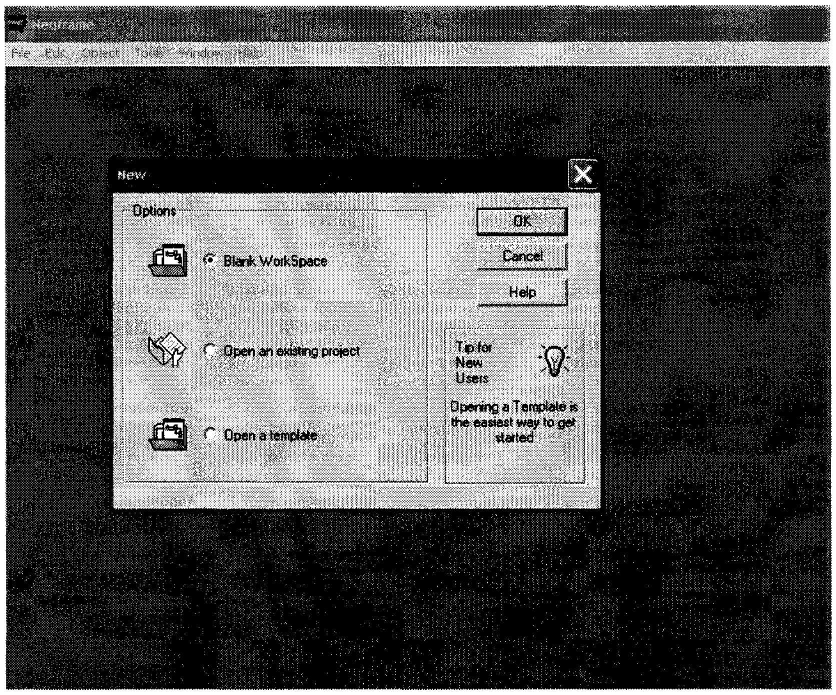
DEVELOPMENT OF BACK-PROPAGATION NEURAL NETWORK SOFTWARE USING NEUFRAME

The NeuFrame simulator provides an option BackProp to create the back-propagation learning based neural network. The procedural steps for creating the neural network model in NeuFrame and using it for machining operations selection by using the methodology described in Chapter 3 are as follows:

- Select from the "New" dialog box a blank workspace as shown in Figure A.1.
- Select, from the Objects Toolbar, the object "BackProp", then drag and place it onto the workspace as shown in Figure A.2.
- Select, from the Objects Toolbar, the different objects, namely "Datasheet", "NeuFuzzy", "Encoder", "Error Graph" then drag and place each of them onto the workspace as shown in Figure A.3. Datasheet1 contains the training data, Datasheet6, Datasheet7 and Datasheet8 contains diameter, tolerance and surface finish values respectively of all the training examples. NeuFuzzy3, NeuFuzzy7 and NeuFuzzy1 contains the rules for classifying the diameter, tolerance and surface finish values of the training examples into different ranges. Datasheet 2 contains the values representing the membership of the above diameter, tolerance and surface finish values in different ranges. Encoder1 is used to normalize the above values to lie in the range from 0 to 1. Network1 stands for the neural network. Datasheet3 contains the query data, Datasheet9, Datasheet10 and Datasheet11 contains diameter, tolerance and surface finish values respectively of all the query examples. NeuFuzzy2, NeuFuzzy4 and NeuFuzzy5 contains the rules for classifying the diameter, tolerance and surface finish values of the query examples into different ranges. Datasheet 4 contains the values representing the membership of the above diameter, tolerance and surface finish values in different ranges. Encoder1 is used to again normalize the above values to lie in

the range from 0 to 1. Datasheet5 contains the output generated by the neural network in response to the query examples.

- Establish the connections between the different objects as shown in Figure A.3.
- Select the number of input, hidden and output layer nodes of the neural network and select the transfer function for each layer as shown in Figure A.4.
- Set the Learning Rate and the Momentum Rate and Set the method of updating weights to either after each epoch or after each pattern as shown in Figure A.5.
- Set the criterion for stopping the training of the neural network to either the epoch count or the training error, and select their appropriate values as shown in Figure A.6.
- Load the training data from the file into the Datasheet1 in the workspace as shown in Figure A.7.
- Set the neural network to training mode and click the "Play" button on the control bar to start the training of the neural network. The training progresses until the training error reaches the default stop error.
- Load the query data from the file into the Datasheet3 for training in the workspace.
- Set the neural network to query mode and click the "Play" button to start processing of the query data.



[✓ Blank WorkSpace] Open an existing project Open a template

Opening a Template is the easiest way to get started

Figure A.1. Creating a Blank Workspace

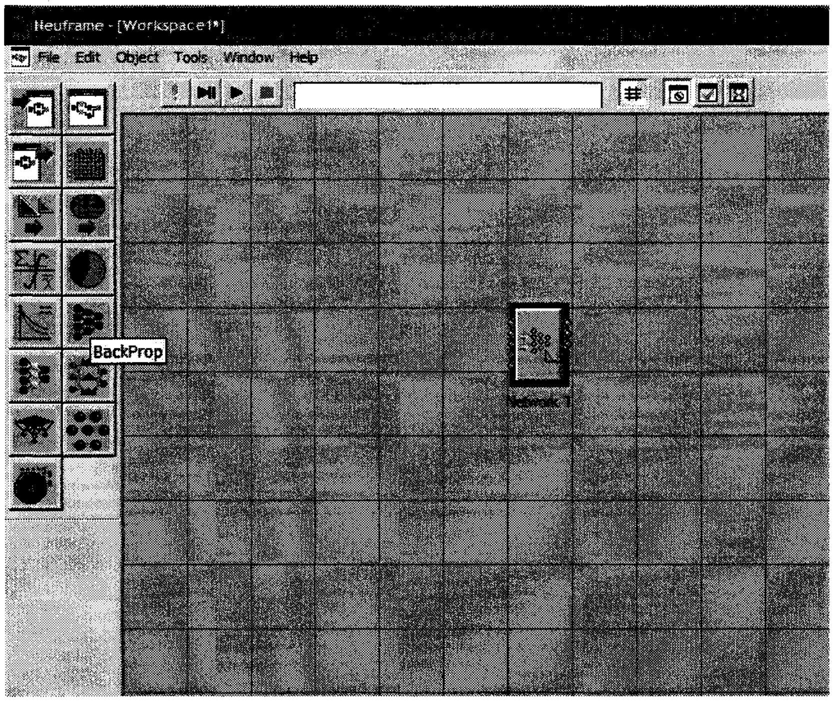


Figure A.2 Selecting the Object "BackProp" and Placing it onto the Workspace

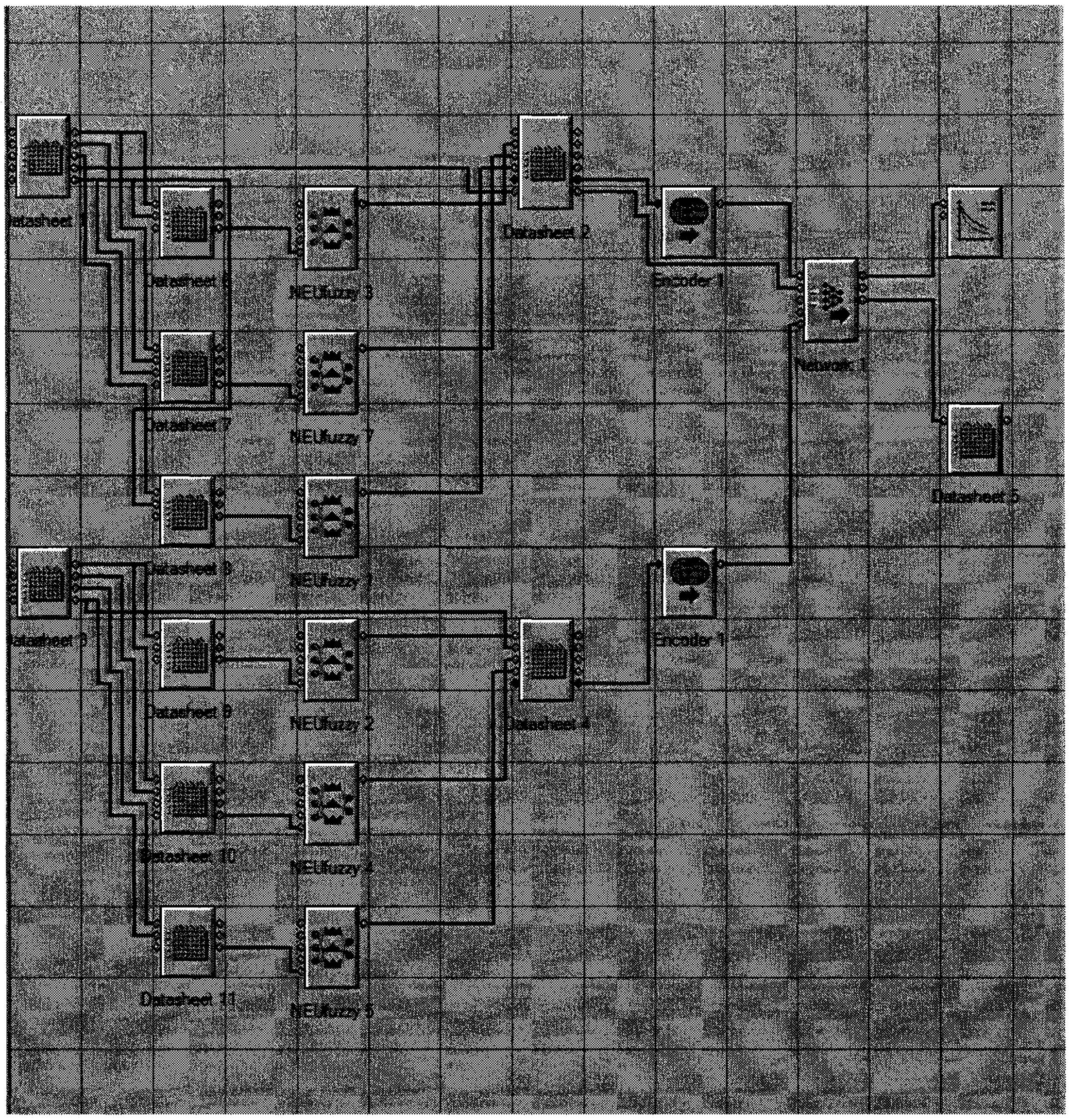
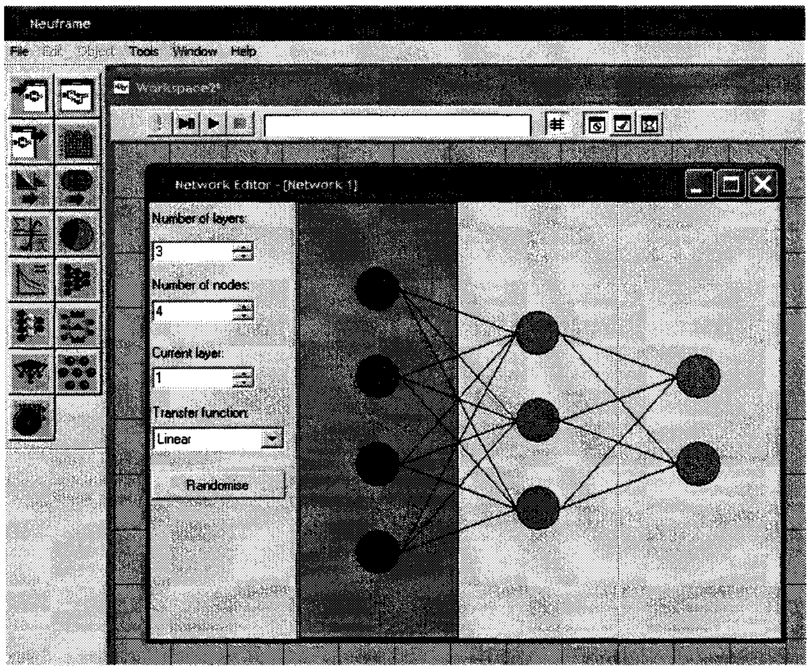


Figure A.3. The NeuFrame Workspace and the Different Connections Between Objects


Neuframe
File Edit Object Tools Window Help
Workspace2*
Network Editor - [Network 1]
Randomise

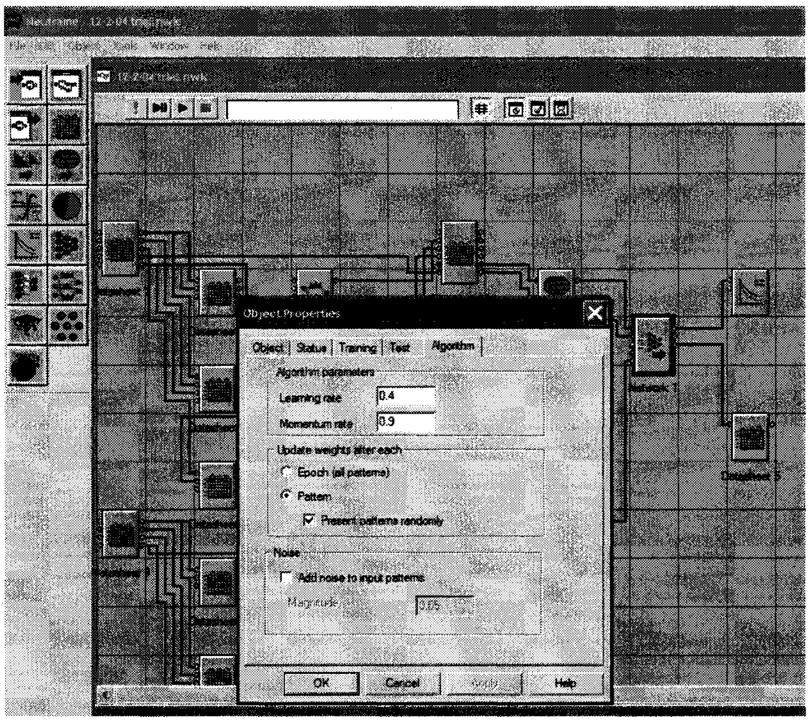
Number of layers: 3

Number of nodes: 4

Current layer: 1

Transfer function: Linear

Figure A.4 Selecting the Input, Hidden and Output Layer Nodes of the Neural Network



Algorithm parameters

Learning rate: 0.4

Momentum rate: 0.9

Update weights after each

( ) Epoch (all patterns)

(•) Pattern

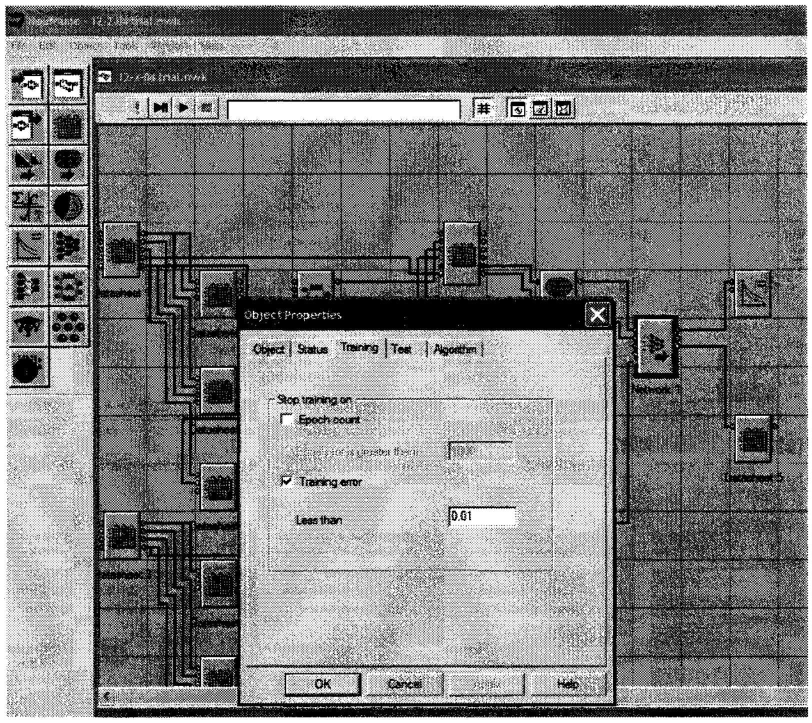
[✓] Present patterns randomly

Noise

[ ] Add noise to input patterns

Magnitude: 0.05

Figure A.5 Selecting the Learning Rate and the Momentum Rate

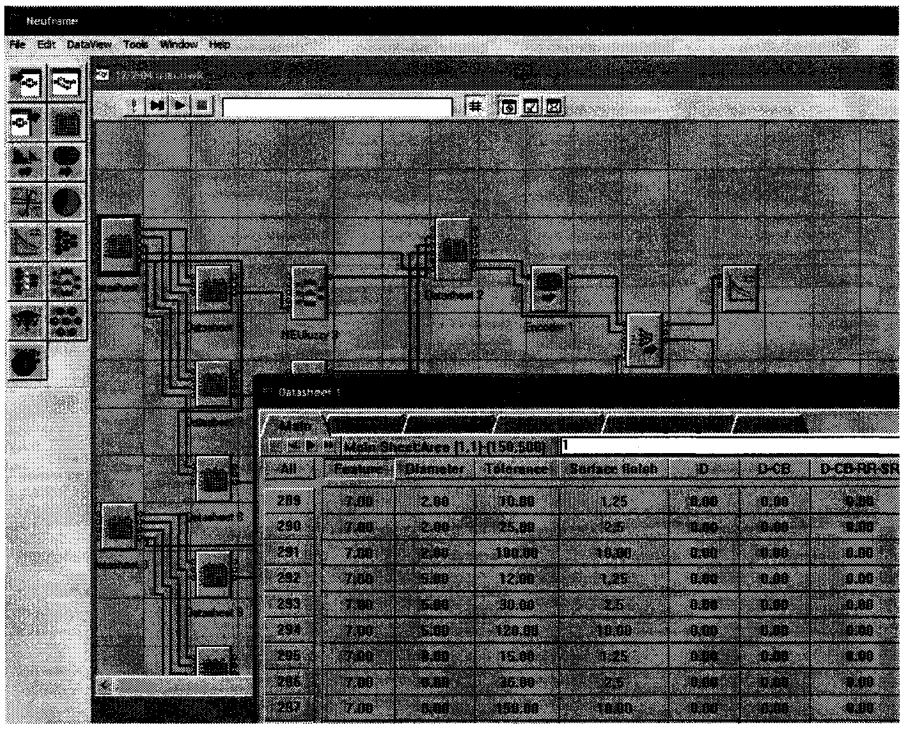


Stop training on

- [ ] Epoch count
- [x] Training error

Less than 0.01

Figure A.6 Selecting the Criterion to Stop the Training of the Neural Network



Main Sheet Area [1.1]-[150,500] 1

All	Feature	Diameter	Tolerance	Surface finish	D	D-CB	D-CB-RR-SR
289	7.00	2.00	10.00	1.25	0.00	0.00	0.00
290	7.00	2.00	25.00	2.5	0.00	0.00	0.00
291	7.00	2.00	100.00	10.00	0.00	0.00	0.00
292	7.00	5.00	12.00	1.25	0.00	0.00	0.00
293	7.00	5.00	30.00	2.5	0.00	0.00	0.00
294	7.00	5.00	120.00	10.00	0.00	0.00	0.00
295	7.00	8.00	15.00	1.25	0.00	0.00	0.00
296	7.00	8.00	36.00	2.5	0.00	0.00	0.00
297	7.00	8.00	150.00	10.00	0.00	0.00	0.00

Figure A.7 Loading the Training Data from the File into the Datasheet1

APPENDIX B

DEVELOPMENT OF EXPERT SYSTEM SOFTWARE USING CLIPS

The program developed by the author in CLIPS to show the implementation of the expert system based methodology for set-up planning described in Chapter 4 and the step by step procedure for executing the program are presented here. The program has been tested using the CLIPS expert system shell version 6.10 compiler under the Windows environment. Although other versions of CLIPS may be used, they have not been fully tested and some changes may be necessary to compile and run this program. The program contains the rules comprising the knowledge base for solving the different set-up planning problems as well as the definitions of templates for different facts and the functions used to perform various calculations as explained in Chapter 4. The program is contained in mainly four files, namely

program_file_1.clp,
program_file_2.clp,
program_file_3.clp, and
program_file_4.clp.

Also another file, namely datafile.clp has been presented here containing the input data to the expert system for set-up planning for the illustrative example considered in Chapter 4.

The program contained in the file, program_file_1.clp is used for for grouping of machining operations into different set-ups and for generation of precedence constraints between operations. The input to the program is the file, datafile.clp which needs to be provided by the user and the output generated by the program is stored in the file, facts1.clp.

The program contained in the file, program_file_2.clp is used to generate a list of all the preceding operations for a given machining operation from the precedence constraints between operations. The input to the program is the file, facts1.clp and the output generated by the program is stored in the file, facts2.clp.

The program contained in the file, program_file_3.clp is used to determine the sequence of operations in each set-up, subject to the precedence constraints generated earlier as well as various manufacturing logic based on heuristic and expert knowledge. The input to the program is the file, facts2.clp and the output generated by the program is the file, facts3.clp.

The program contained in the file, program_file_4.clp is used to determine the locating and clamping surfaces for each set-up. The input to the program is the file, facts1.clp and the output generated by the program is stored in the file, facts4.clp.

The procedural setps for executing a program file in CLIPS are as follows:

- Load the program from the file into the CLIPS environment by selecting from the File Menu "Load CLIPS Constructs".
- Load the input data from the datafile into the CLIPS environment by selecting from the File Menu "Load CLIPS Constructs".
- Excute the program by first by selecting from the Execution Menu "Reset" and then "Run".

The sequence of execution of the different program files as well as the files containing the input and output from each program file is schematically shown in Figure B.1.

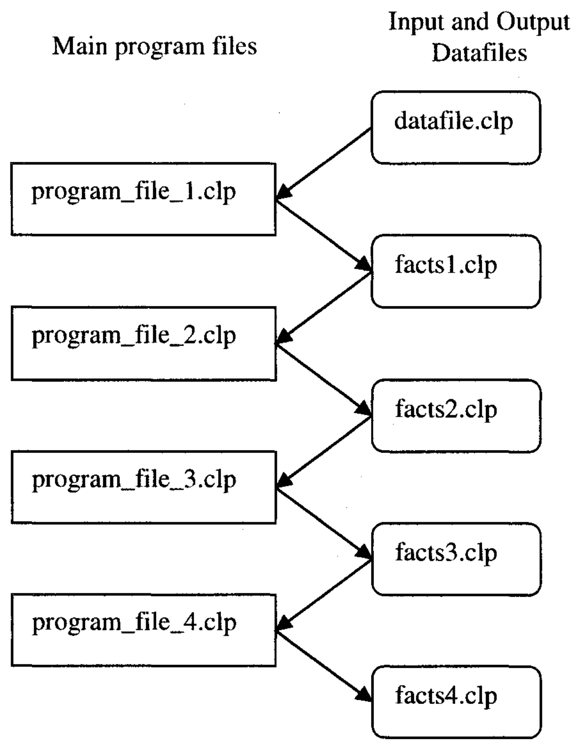


Figure B.1 Sequence of Execution of the Different Program Files of the Expert System

The program developed by the author in CLIPS for solving the different set-up planning problems is given as follows.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; file name: program_file_1.CLP;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;==================================================================================
;;;     Rules for set-up formation and for generation of precedence constraints between operations
;;;     CLIPS Version 6.10
;;;    To execute, merely load this program, then load the file,"datafile.clp", then reset and run.
;;;==================================================================================
(defmodule MAIN
 (export deftemplate ?ALL))

;;;****************************
;;;* DEFTEMPLATE DEFINITIONS *
;;;****************************

;;; The feature facts hold the information on each feature such as identification number,
;;; name, type,sub-type, adjacent features, reference/datum features, various dimensions,
;;; tool access directions (TAD), etc.

 (deftemplate MAIN::feature
            (slot number (type INTEGER) (default ?NONE))
            (slot name (type SYMBOL) (allowed-symbols CHAMFER EXTERNAL_STEP FACE GROOVE HOLE KEYWAY
EXTERNAL_TAPER THREAD HOLE))
            (slot type (type SYMBOL) (allowed-symbols EXTERNAL INTERNAL))
            (slot subtype (type SYMBOL) (allowed-symbols PRIMARY SECONDARY))
            (multislot secondary_feature_to (type INTEGER) (default 0))
            (multislot adjacent_features (type INTEGER) (default ?DERIVE))
            (multislot adjacent_features_names (type SYMBOL) (allowed-symbols CHAMFER EXTERNAL_STEP FACE
GROOVE HOLE KEYWAY EXTERNAL_TAPER THREAD HOLE))
            (multislot reference_features (type INTEGER) (default 0))
            (slot step_diameter (type NUMBER))
            (multislot adjacent_step_diameters (type NUMBER))
            (slot hole_diameter (type NUMBER))
            (slot hole_depth (type NUMBER))
            (multislot adjacent_hole_diameters (type NUMBER))
            (multislot adjacent_hole_depths (type NUMBER))
            (slot TAD (type SYMBOL) (allowed-symbols left right right-left) (default ?NONE)))

;;; The operation facts hold the information on each machining operation such as identification number, type, machining stage,
;;; feature to be machined, geometric tolerance relationships with other features, tool access directions (TAD), etc.

(deftemplate MAIN::operation
            (slot number (type INTEGER) (default ?NONE))
            (slot type (type SYMBOL) (default ?NONE))
            (slot machining_stage (type SYMBOL) (allowed-symbols rough semifinish finish) (default rough))
            (slot on-feature (type INTEGER) (default ?NONE))
            (slot TAD (type SYMBOL) (allowed-symbols left right right-left) (default ?NONE))
            (multislot relation-with-feature (type INTEGER) (default 0))
            (multislot tolerance (type NUMBER) (default 0)))
(deftemplate MAIN::setup-from-left-cluster (multislot operation_numbers))
(deftemplate MAIN::setup-from-right-cluster (multislot operation_numbers))
(deftemplate MAIN::order-of-features-from-left-external-to-internal
            (multislot feature-numbers (type INTEGER) (default ?NONE)))
(deftemplate MAIN::order-of-features-from-right-external-to-internal
            (multislot feature-numbers (type INTEGER) (default ?NONE)))
(deftemplate MAIN::feature-with-largest-dia
            (slot number (type INTEGER) (default ?NONE)))

;;;***************************************************************************************
;;;* DEFFUNCTION DEFINITIONS USED IN THE RULES FOR GENERATING PRECEDENCE CONSTRAINTS*
;;;***************************************************************************************

;;; This function is used to return the order of precedence between features based on machining
;;; of reference features first

(deffunction MAIN::function1 (?f1 ?a)
            (bind $?num (fact-slot-value ?f1 reference_features))
            (while (>= (length$ $?num) 1)
            (bind ?num1 (first$ $?num))
            (bind $?num (rest$ $?num))
            (assert (feature_precedence ?num1 ?a))))

;;;****************************************************************
;;;* RULES FOR GENERATING PRECEDENCE CONSTRAINTS BETWEEN FEATURES*
;;;****************************************************************
;;; The following rules are used to establish the precedence constraints between features based on machining from left to right

(defrule MAIN::precedence_constraint_according_to_order_left-to-right-1
            (feature (number ?a) (type EXTERNAL) (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER|GROOVE) (TAD
left))
            (feature (number ?b) (type EXTERNAL) (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER|GROOVE) (TAD
left))
            (order-of-features-from-left-external-to-internal (feature-numbers $? ?a $? ?b $?))
=>          (assert (feature_precedence ?a ?b)))
(defrule MAIN::precedence_constraint_according_to_order_left-to-right-2
            (feature (number ?a) (type EXTERNAL) (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER|GROOVE) (TAD
left))
            (feature (number ?b) (type EXTERNAL) (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER|GROOVE) (TAD
?TAD&left|right-left))
            (order-of-features-from-left-external-to-internal (feature-numbers $? ?a $? ?b $?))
=>          (assert (feature_precedence ?a ?b)))
(defrule MAIN::precedence_constraint_according_to_order_left-to-right-3
            (feature (number ?a) (type EXTERNAL) (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER|GROOVE) (TAD
?TAD&left|right-left))
            (feature (number ?b) (type EXTERNAL) (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER|GROOVE) (TAD
left))
            (order-of-features-from-left-external-to-internal (feature-numbers $? ?a $? ?b $?))
=>          (assert (feature_precedence ?a ?b)))
(defrule MAIN::precedence_constraint_according_to_order_left-to-right-4
            (feature (number ?a) (type EXTERNAL) (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER|GROOVE) (TAD
?TAD&left|right-left))
            (feature (number ?b) (type EXTERNAL) (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER|GROOVE) (TAD
?TAD&left|right-left))
            (order-of-features-from-left-external-to-internal (feature-numbers $? ?a $? ?b $?))
=>          (assert (feature_precedence ?a ?b)))

;;; The following rules are used to establish the precedence constraints between features based on machining from right to left

(defrule MAIN::precedence_constraint_according_to_order-right-to-left-1
            (feature (number ?a) (type EXTERNAL) (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER|GROOVE) (TAD
right))
            (feature (number ?b) (type EXTERNAL)  (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER|GROOVE) (TAD
right))
            (order-of-features-from-right-external-to-internal (feature-numbers $? ?a $? ?b $?))
=>          (assert (feature_precedence ?a ?b)))
(defrule MAIN::precedence_constraint_according_to_order-right-to-left-2
            (feature (number ?a) (type EXTERNAL) (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER|GROOVE) (TAD
right))
            (feature (number ?b) (type EXTERNAL)  (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER|GROOVE) (TAD
?TAD&right|right-left))
            (order-of-features-from-right-external-to-internal (feature-numbers $? ?a $? ?b $?))
=>          (assert (feature_precedence ?a ?b)))
(defrule MAIN::precedence_constraint_according_to_order-right-to-left-3
            (feature (number ?a) (type EXTERNAL) (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER|GROOVE) (TAD
?TAD&right|right-left))
            (feature (number ?b) (type EXTERNAL)  (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER|GROOVE) (TAD
right))
            (order-of-features-from-right-external-to-internal (feature-numbers $? ?a $? ?b $?))
=>          (assert (feature_precedence ?a ?b)))
(defrule MAIN::precedence_constraint_according_to_order-right-to-left-4
            (feature (number ?a) (type EXTERNAL) (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER|GROOVE) (TAD
?TAD&right|right-left))
            (feature (number ?b) (type EXTERNAL)  (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER|GROOVE) (TAD
?TAD&right|right-left))
```

```
        (order-of-features-from-right-external-to-internal (feature-numbers $? ?a $? ?b $?))
=>      (assert (feature_precedence ?a ?b)))

;;; The following rules are used to establish the various precedence constraints between machining operations on features
;;; based on heuristic and expert knowledge

(defrule MAIN::precedence_constraint_25 "precedence based on machining of external steps of larger dia first"
        ?f1 <- (feature (number ?a) (name EXTERNAL_STEP) (step_diameter ?d1) (adjacent_features ?b ?c)
(adjacent_features_names EXTERNAL_STEP EXTERNAL_STEP) (adjacent_step_diameters ?d2 ?d3))
        (test (and (< ?d1 ?d2) (< ?d1 ?d3)))
        (feature (number ?d) (type EXTERNAL) (name EXTERNAL_STEP))
        ?f2 <- (feature_precedence ?a ?d)
=>      (retract ?f2)
        (assert (feature_precedence ?b ?a))
        (assert (feature_precedence ?c ?a))
        (assert (feature_precedence ?d ?a)))
(defrule MAIN::precedence_constraint_l "precedence based on machining of end faces first"
        (feature (number ?a) (name FACE) (type EXTERNAL) (adjacent_features ? ?) (adjacent_features_names
EXTERNAL_STEP HOLE)(TAD left))
        (feature (number ?b) (TAD ?TAD&left|right-left))
        (test (not (= ?a ?b)))
=>      (assert (feature_precedence ?a ?b)))
(defrule MAIN::precedence_constraint_r "precedence based on machining of end faces first"
        (feature (number ?a) (name FACE) (type EXTERNAL) (adjacent_features ? ?) (adjacent_features_names
EXTERNAL_STEP HOLE)(TAD right))
        (feature (number ?b) (TAD ?TAD&right|right-left))
        (test (not (= ?a ?b)))
=>      (assert (feature_precedence ?a ?b)))
(defrule MAIN::precedence_constraint_lnew1 "precedence based on machining of end faces first"
        (feature (number ?a) (name FACE) (type EXTERNAL) (adjacent_features ? ?) (adjacent_features_names THREAD
HOLE)(TAD left))
        (feature (number ?b) (TAD ?TAD&left|right-left))
        (test (not (= ?a ?b)))
=>      (assert (feature_precedence ?a ?b)))
(defrule MAIN::precedence_constraint_rnew1 "precedence based on machining of end faces first"
        (feature (number ?a) (name FACE) (type EXTERNAL) (adjacent_features ? ?) (adjacent_features_names THREAD
HOLE)(TAD right))
        (feature (number ?b) (TAD ?TAD&right|right-left))
        (test (not (= ?a ?b)))
=>      (assert (feature_precedence ?a ?b)))
(defrule MAIN::precedence_constraint_lnew "precedence based on machining of end faces first"
        (feature (number ?a) (name FACE) (type EXTERNAL) (adjacent_features ?)(TAD left))
        (feature (number ?b) (TAD ?TAD&left|right-left))
        (test (not (= ?a ?b)))
=>      (assert (feature_precedence ?a ?b)))
(defrule MAIN::precedence_constraint_rnew "precedence based on machining of end faces first"
        (feature (number ?a) (name FACE) (type EXTERNAL) (adjacent_features ?)(TAD right))
        (feature (number ?b) (TAD ?TAD&right|right-left))
        (test (not (= ?a ?b)))
=>      (assert (feature_precedence ?a ?b)))
(defrule MAIN::precedence_constraint_l1 "precedence based on machining of end faces first"
        ?f1 <- (feature (number ?a) (name FACE) (type EXTERNAL) (adjacent_features ?b)(TAD left))
        (feature (number ?b) (TAD ?TAD&left|right-left))
        (test (not (= ?a ?b)))
=>      (assert (feature_precedence ?a ?b)))
(defrule MAIN::precedence_constraint_r1 "precedence based on machining of end faces first"
        ?f1 <- (feature (number ?a) (name FACE) (type EXTERNAL) (adjacent_features ?b)(TAD right))
        (feature (number ?b) (TAD ?TAD&right|right-left))
        (test (not (= ?a ?b)))
=>      (assert (feature_precedence ?a ?b)))
(defrule MAIN::precedence_constraint_1_locating "precedence based on machining of reference features first"
        ?f1 <-(feature (number ?a) (reference_features $?b))
        (test (> (nth$ 1 (explode$ (implode$ (first$ (fact-slot-value ?f1 reference_features))))) 0))
=>      (function1 ?f1 ?a))
(defrule MAIN::precedence_constraint_2_locating "precedence based on machining of end faces first"
        ?f1 <- (feature (number ?a) (name FACE) (type EXTERNAL) (adjacent_features ?b))
```

```
            (test (= (length$ (fact-slot-value ?f1 adjacent_features)) 1))
=>          (assert (feature_precedence ?a ?b)))
(defrule MAIN::precedence_constraint_3_locating "precedence based on machining of end faces first"
            ?f1 <- (feature (number ?a) (name FACE) (type EXTERNAL) (adjacent_features ?b ?c) (adjacent_features_names
EXTERNAL_STEP HOLE))
=>          (assert (feature_precedence ?a ?b))
            (assert (feature_precedence ?a ?c)))
(defrule MAIN::precedence_constraint_4_locating "precedence based on machining of end faces first"
            ?f1 <- (feature (number ?a) (name FACE) (type EXTERNAL) (adjacent_features ?b ?c) (adjacent_features_names HOLE
EXTERNAL_STEP))
=>          (assert (feature_precedence ?a ?b))
            (assert (feature_precedence ?a ?c)))
(defrule MAIN::precedence_constraint_5_accessibility "precedence based on machining of adjacent external steps first prior to that
of the face between them"
            ?f1 <- (feature (number ?a) (name FACE) (type EXTERNAL) (adjacent_features ?b ?c) (adjacent_features_names
EXTERNAL_STEP EXTERNAL_STEP))
            (test (= (length$ (fact-slot-value ?f1 adjacent_features)) 2))
=>          (assert (feature_precedence ?b ?a))
            (assert (feature_precedence ?c ?a)))
(defrule MAIN::precedence_constraint_6_nondestruction "precedence based on machining of a chamfer prior to that of the adjacent
thread"
            (feature (number ?a) (name THREAD) (type EXTERNAL) (secondary_feature_to ?b) (adjacent_features $? ?c)
(adjacent_features_names $? CHAMFER))
=>          (assert (feature_precedence ?b ?a))
            (assert (feature_precedence ?c ?a))
            (assert (feature_precedence ?b ?c)))
(defrule MAIN::precedence_constraint_7_nondestruction "precedence based on machining of a chamfer prior to that of the adjacent
thread"
            (feature (number ?a) (name THREAD) (type EXTERNAL) (secondary_feature_to ?b) (adjacent_features ?c $?)
(adjacent_features_names CHAMFER $?))
=>          (assert (feature_precedence ?b ?a))
            (assert (feature_precedence ?c ?a))
            (assert (feature_precedence ?b ?c)))

(defrule MAIN::precedence_constraint_8_nondestruction "precedence based on machining of a groove prior to that of the adjacent
thread"
            (feature (number ?a) (name THREAD) (type EXTERNAL) (secondary_feature_to ?b) (adjacent_features $? ?c $?)
(adjacent_features_names $? GROOVE $?))
=>          (assert (feature_precedence ?c ?a)))

(defrule MAIN::precedence_constraint_9_accessibility "precedence based on machining of adjacent external steps first prior to that
of the groove between them"
            (feature (number ?a) (name GROOVE) (type EXTERNAL) (adjacent_features ?b ?c) (adjacent_features_names
EXTERNAL_STEP EXTERNAL_STEP))
=>          (assert (feature_precedence ?b ?a))
            (assert (feature_precedence ?c ?a)))
(defrule MAIN::precedence_constraint_10_accessibility "precedence based on machining of an external step first prior to that of the
adjacent groove"
            (feature (number ?a) (name GROOVE) (type EXTERNAL) (adjacent_features $? ?b) (adjacent_features_names $?
EXTERNAL_STEP))
=>          (assert (feature_precedence ?b ?a)))
(defrule MAIN::precedence_constraint_11_accessibility "precedence based on machining of an external step first prior to that of the
adjacent groove"
            (feature (number ?a) (name GROOVE) (type EXTERNAL) (adjacent_features ?b $?) (adjacent_features_names
EXTERNAL_STEP $?))
=>          (assert (feature_precedence ?b ?a)))
(defrule MAIN::precedence_constraint_12_accessibility "precedence based on machining of an external step first prior to that of the
adjacent taper"
            (feature (number ?a) (name EXTERNAL_TAPER) (type EXTERNAL) (adjacent_features $? ?b $?)
(adjacent_features_names $? EXTERNAL_STEP $?))
=>          (assert (feature_precedence ?b ?a)))
(defrule MAIN::precedence_constraint_13_nondestruction "precedence based on machining of the external step and the face prior to
that of the chamfer between them"
            ?f1 <- (feature (number ?a) (name CHAMFER) (type EXTERNAL) (adjacent_features ?b ?c) (adjacent_features_names
EXTERNAL_STEP FACE))
=>          (assert (feature_precedence ?b ?a))
```

```
            (assert (feature_precedence ?c ?a)))
(defrule MAIN::precedence_constraint_14_nondestruction "precedence based on machining of the external step and the face prior to
that of the chamfer between them"
            ?f1 <- (feature (number ?a) (name CHAMFER) (type EXTERNAL) (adjacent_features ?b ?c) (adjacent_features_names
FACE EXTERNAL_STEP))
=>          (assert (feature_precedence ?b ?a))
            (assert (feature_precedence ?c ?a)))
(defrule MAIN::precedence_constraint_15 "precedence based on machining of a face prior to that of the adjacent chamfer"
            ?f1 <- (feature (number ?a) (name CHAMFER) (type EXTERNAL) (adjacent_features ? ?b) (adjacent_features_names ?
FACE))
=>          (assert (feature_precedence ?b ?a)))
(defrule MAIN::precedence_constraint_16 "precedence based on machining of a face prior to that of the adjacent chamfer"
            ?f1 <- (feature (number ?a) (name CHAMFER) (type EXTERNAL) (adjacent_features ?b ?) (adjacent_features_names
FACE ?))
=>          (assert (feature_precedence ?b ?a)))
(defrule MAIN::precedence_constraint_17_accessibility "precedence based on machining of the end face prior to that of the adjacent
hole"
            ?f1 <- (feature (number ?a) (name HOLE) (adjacent_features ?b) (adjacent_features_names FACE))
            (test (= (length$ (fact-slot-value ?f1 adjacent_features)) 1))
=>          (assert (feature_precedence ?b ?a)))
(defrule MAIN::precedence_constraint_18_accessibility "precedence based on machining of the end face and the hole of smaller dia
prior to that of the adjacent coaxial hole of larger dia"
            ?f1 <- (feature (number ?a) (name HOLE) (hole_diameter ?d1) (hole_depth ?h1) (adjacent_features ?b ?c)
(adjacent_features_names FACE HOLE) (adjacent_hole_diameters ?d2) (adjacent_hole_depths ?h2))
            (test (> ?d1 ?d2))
            (test (< ?h1 ?h2))
=>          (assert (feature_precedence ?b ?a))
            (assert (feature_precedence ?c ?a)))
(defrule MAIN::precedence_constraint_19_accessibility "precedence based on machining of the end face and the hole of smaller dia
prior to that of the adjacent coaxial hole of larger dia"
            ?f1 <- (feature (number ?a) (name HOLE) (hole_diameter ?d1) (hole_depth ?h1) (adjacent_features ?b ?c)
(adjacent_features_names HOLE FACE) (adjacent_hole_diameters ?d2) (adjacent_hole_depths ?h2))
            (test (> ?d1 ?d2))
            (test (< ?h1 ?h2))
=>          (assert (feature_precedence ?c ?a))
            (assert (feature_precedence ?b ?a)))
(defrule MAIN::precedence_constraint_20 "precedence based on machining of the hole of smaller dia prior to that of the adjacent
coaxial hole of larger dia"
            ?f1 <- (feature (number ?a) (name HOLE) (hole_diameter ?d1) (hole_depth ?h1) (adjacent_features ? ?b)
(adjacent_features_names HOLE HOLE) (adjacent_hole_diameters ? ?d2) (adjacent_hole_depths ? ?h2))
            (test (> ?d1 ?d2))
=>          (assert (feature_precedence ?b ?a)))
(defrule MAIN::precedence_constraint_21 "precedence based on machining of the hole of smaller dia prior to that of the adjacent
coaxial hole of larger dia"
            ?f1 <- (feature (number ?a) (name HOLE) (hole_diameter ?d1) (hole_depth ?h1) (adjacent_features ?b ?)
(adjacent_features_names HOLE HOLE) (adjacent_hole_diameters ?d2 ?) (adjacent_hole_depths ?h2 ?))
            (test (> ?d1 ?d2))
=>          (assert (feature_precedence ?b ?a)))
(defrule MAIN::precedence_constraint_22 "precedence based on machining of the hole of smaller dia prior to that of the adjacent
coaxial hole of larger dia"
            ?f1 <- (feature (number ?a) (name HOLE) (hole_diameter ?d1) (hole_depth ?h1) (adjacent_features ? ?b)
(adjacent_features_names HOLE HOLE) (adjacent_hole_diameters ? ?d2) (adjacent_hole_depths ? ?h2))
            (test (< ?d1 ?d2))
=>          (assert (feature_precedence ?a ?b)))
(defrule MAIN::precedence_constraint_23 "precedence based on machining of the hole of smaller dia prior to that of the adjacent
coaxial hole of larger dia"
            ?f1 <- (feature (number ?a) (name HOLE) (hole_diameter ?d1) (hole_depth ?h1) (adjacent_features ?b ?)
(adjacent_features_names HOLE HOLE) (adjacent_hole_diameters ?d2 ?) (adjacent_hole_depths ?h2 ?))
            (test (< ?d1 ?d2))
=>          (assert (feature_precedence ?a ?b)))
(defrule MAIN::precedence_constraint_24 "precedence based on machining of a hole of smaller dia prior to that of the adjacent hole
of larger dia"
            ?f1 <- (feature (number ?a) (name HOLE) (hole_diameter ?d1) (hole_depth ?h1) (adjacent_features ?b)
(adjacent_features_names HOLE) (adjacent_hole_diameters ?d2) (adjacent_hole_depths ?h2))
            (test (< ?d1 ?d2))
            (test (> ?h1 ?h2))
```

```
=>          (assert (feature_precedence ?a ?b)))
(defrule MAIN::precedence_constraint_27 "precedence based on machining of the hole of smallest dia prior to that of the adjacent
hole of larger dia"
            ?f1 <- (feature (number ?a) (name HOLE) (hole_diameter ?d1) (hole_depth ?h1) (adjacent_features ? ?b)
(adjacent_features_names ? HOLE) (adjacent_hole_diameters ?d2) (adjacent_hole_depths ?h2))
            ?f2 <- (feature (number ?b) (name HOLE) (hole_diameter ?d2) (hole_depth ?h2) (adjacent_features ?a ?c)
(adjacent_features_names HOLE HOLE) (adjacent_hole_diameters ?d1 ?d3) (adjacent_hole_depths ?h1 ?h3))
            (test (and (< ?d2 ?d1) (< ?d3 ?d2) (< ?d3 ?d1)))
=>          (assert (feature_precedence ?c ?a)))
(defrule MAIN::precedence_constraint_28_nondestruction "precedence based on machining of primary features prior to that of
secondary features such as keyway"
            (feature (number ?a) (name KEYWAY) (type EXTERNAL) (secondary_feature_to $? ?b $?))
=>
            (assert (feature_precedence ?b ?a)))

;;;*****************************************************************
;;;* DEFFUNCTION DEFINITIONS USED IN THE RULES FOR SETUP FORMATION*
;;;*****************************************************************
;;; The following functions are used to return the identifier of the feature with which a given feature, on which
;;; the machining operation is to be performed, has the tightest tolerance relationship

(deffunction MAIN::feature-with-tightest-tolerance (?f1)
            (bind $?num (fact-slot-value ?f1 tolerance))
            (while (>= (length$ $?num) 2)
              (bind ?num1 (first$ $?num))
              (bind $?num (rest$ $?num))
              (bind ?num2 (first$ $?num))
              (bind ?num11 (nth$ 1 (explode$ (implode$ ?num1))))
              (bind ?num22 (nth$ 1 (explode$ (implode$ ?num2))))
              (if (> ?num11 ?num22)
                then (bind ?num1 ?num2)
              else (bind ?num1 ?num1)))
            (bind ?num3 (member$ (explode$ (implode$ ?num1)) (fact-slot-value ?f1 tolerance)))
            (bind ?num4 (nth$ ?num3 (fact-slot-value ?f1 relation-with-feature)))
            return ?num4)
(deffunction MAIN::feature-with-tightest-tolerance-case-of-one-feature (?f1)
            (bind ?num5 (nth$ 1 (explode$ (implode$ (fact-slot-value ?f1 relation-with-feature)))))
            return ?num5)
(deffunction MAIN::feature-with-tightest-tolerance-0 (?f1)
            (bind $?num (fact-slot-value ?f1 tolerance))
            (while (>= (length$ $?num) 2)
              (bind ?num1 (first$ $?num))
              (bind $?num (rest$ $?num))
              (bind ?num2 (first$ $?num))
              (bind ?num11 (nth$ 1 (explode$ (implode$ ?num1))))
              (bind ?num22 (nth$ 1 (explode$ (implode$ ?num2))))
              (if (> ?num11 ?num22)
                then (bind ?num1 ?num2)
              else (bind ?num1 ?num1)))
            (bind ?num3 (member$ (explode$ (implode$ ?num1)) (fact-slot-value ?f1 tolerance)))
            return ?num3)
(deffunction MAIN::feature-with-tightest-tolerance-1 (?f1)
            (bind ?num5 (feature-with-tightest-tolerance-0 ?f1))
            (bind $?num6 (fact-slot-value ?f1 tolerance))
            (bind $?num (delete$ $?num6 ?num5 ?num5))
            (bind ?num1 (first$ $?num))
            (while (>= (length$ $?num) 2)
              (bind ?num1 (first$ $?num))
              (bind $?num (rest$ $?num))
              (bind ?num2 (first$ $?num))
              (bind ?num11 (nth$ 1 (explode$ (implode$ ?num1))))
              (bind ?num22 (nth$ 1 (explode$ (implode$ ?num2))))
              (if (> ?num11 ?num22)
                then (bind ?num1 ?num2)
              else (bind ?num1 ?num1)))
            (bind ?num3 (member$ (explode$ (implode$ ?num1)) (fact-slot-value ?f1 tolerance)))
            (bind ?num4 (nth$ ?num3 (fact-slot-value ?f1 relation-with-feature)))
            return ?num4)

;;; The following functions are used to update the tolerance relationship vectors by removing those
;;; tolerance relationships between features that have been already satisfied

(deffunction MAIN::update-relation-with-feature (?f1)
            (bind ?num (fact-slot-value ?f1 relation-with-feature))
            (bind ?num1 (feature-with-tightest-tolerance ?f1))
            (bind ?num2 (delete-member$ (create$ ?num) ?num1))
            (return ?num2))
(deffunction MAIN::update-relation-with-feature-1 (?f1)
            (bind ?num (fact-slot-value ?f1 relation-with-feature))
            (bind ?num1 (feature-with-tightest-tolerance-1 ?f1))
            (bind ?num2 (delete-member$ (create$ ?num) ?num1))
            (return ?num2))

;;; The following function is used to return the identifier of the feature with which a given feature, on which
;;; the machining operation is to be performed, has the has the tightest tolerance relationship

(deffunction MAIN::feature-with-tightest-tolerance00 (?f1)
            (bind $?num (fact-slot-value ?f1 tolerance))
            (while (>= (length$ $?num) 2)
              (bind ?num1 (first$ $?num))
              (bind $?num (rest$ $?num))
              (bind ?num2 (first$ $?num))
              (bind ?num11 (nth$ 1 (explode$ (implode$ ?num1))))
              (bind ?num22 (nth$ 1 (explode$ (implode$ ?num2))))
              (if (> ?num11 ?num22)
                then (bind ?num1 ?num2)
              else (bind ?num1 ?num1)))
            (return ?num1))

;;; The following function is used to update the tolerance relationship vectors by removing those
;;; tolerance relationships between features that have been already satisfied

(deffunction MAIN::update-tolerance (?f1)
            (bind ?num (fact-slot-value ?f1 tolerance))
            (bind ?num1 (feature-with-tightest-tolerance00 ?f1))
            (bind ?num2 (delete-member$ (create$ ?num) ?num1))
            (return ?num2))


;;; The following function is used to return the identifier of the feature with which a given feature, on which
;;; the machining operation is to be performed, has the has the tightest tolerance relationship

(deffunction MAIN::feature-with-tightest-tolerance11 (?f1)
            (bind ?num5 (feature-with-tightest-tolerance-0 ?f1))
            (bind $?num6 (fact-slot-value ?f1 tolerance))
            (bind $?num (delete$ $?num6 ?num5 ?num5))
            (bind ?num1 (first$ $?num))
            (while (>= (length$ $?num) 2)
              (bind ?num1 (first$ $?num))
              (bind $?num (rest$ $?num))
              (bind ?num2 (first$ $?num))
              (bind ?num11 (nth$ 1 (explode$ (implode$ ?num1))))
              (bind ?num22 (nth$ 1 (explode$ (implode$ ?num2))))
              (if (> ?num11 ?num22)
                then (bind ?num1 ?num2)
              else (bind ?num1 ?num1)))
            (return ?num1))

;;; The following function is used to update the tolerance relationship vectors by removing those
;;; tolerance relationships between features that have been already satisfied

(deffunction MAIN::update-tolerance-1 (?f1)
            (bind ?num (fact-slot-value ?f1 tolerance))
            (bind ?num1 (feature-with-tightest-tolerance11 ?f1))
            (bind ?num2 (delete-member$ (create$ ?num) ?num1))
            (return ?num2))

;;;*****************************
;;;* RULES FOR SETUP FORMATION *
;;;*****************************
;;; The following rules are used to assign a specific TAD to operations on features with multiple TAD's
;;; but no specified tolerance relationship with any other feature

(defrule MAIN::operation-on-feature-with-multiple-TAD-no-relation-case1
            ?f <- (operation (on-feature ?on-feature) (TAD right-left) (relation-with-feature 0))
            ?f1 <- (feature (number ?on-feature))
            (not (operation (TAD ?) (relation-with-feature $? ?on-feature $?)))
=>          (modify ?f (TAD left))
            (modify ?f1 (TAD left)))
(defrule MAIN::operation-on-feature-with-multiple-TAD-no-relation-case1a
            ?f1 <- (operation (on-feature ?on-feature) (TAD right-left) (relation-with-feature 0))
            ?f3 <- (feature (number ?on-feature))
            ?f2<-(operation (on-feature ?on-feature1)(TAD right-left) (relation-with-feature ?on-feature))
            ?f4 <- (feature (number ?on-feature1))
=>          (modify ?f1 (TAD left))
            (modify ?f2 (TAD left))
            (modify ?f3 (TAD left))
            (modify ?f4 (TAD left)))
(defrule MAIN::operation-on-feature-with-multiple-TAD-no-relation-case1b
            ?f <- (operation (on-feature ?on-feature) (TAD right-left) (relation-with-feature 0))
            ?f1 <- (feature (number ?on-feature))
            (operation (TAD left) (relation-with-feature $? ?on-feature $?))
=>          (modify ?f (TAD left))
            (modify ?f1 (TAD left)))
(defrule MAIN::operation-on-feature-with-multiple-TAD-no-relation-case1c
            ?f <- (operation (on-feature ?on-feature) (TAD right-left) (relation-with-feature 0))
            ?f1 <- (feature (number ?on-feature))
            (operation (TAD right) (relation-with-feature $? ?on-feature $?))
=>          (modify ?f (TAD right))
            (modify ?f1 (TAD right)))
(defrule MAIN::operation-on-feature-with-multiple-TAD-no-relation-case1d
            ?f <- (operation (on-feature ?on-feature) (TAD right-left) (relation-with-feature 0))
            ?f1 <- (feature (number ?on-feature))
            (operation (TAD left) (relation-with-feature $? ?on-feature $?) (tolerance $? ?tolerance1 $?))
            (operation (TAD right) (relation-with-feature $? ?on-feature $?) (tolerance $? ?tolerance2 $?))
=>          (if (< ?tolerance1 ?tolerance2)
            then (modify ?f (TAD left))
               (modify ?f1 (TAD left))
            else (modify ?f (TAD right))
               (modify ?f1 (TAD right))))

;;; The following rules are used to assign a specific TAD to operations on features with multiple TAD's
;;; and with tolerance relationships with one or more features

(defrule MAIN::operation-on-feature-with-multiple-TAD-relation-with-one-feature-case1a
            ?f1 <- (operation (TAD right-left)(on-feature ?on-feature))
            ?f2 <- (feature (number ?on-feature))
            (test (= (length$ (fact-slot-value ?f1 tolerance)) 1))
            (operation (TAD left) (on-feature =(feature-with-tightest-tolerance-case-of-one-feature ?f1)))
=>          (modify ?f1 (TAD left) (relation-with-feature 0) (tolerance 0))
            (modify ?f2 (TAD left)))
(defrule MAIN::operation-on-feature-with-multiple-TAD-relation-with-multiple-features-case1
            ?f1 <- (operation (TAD right-left)(on-feature ?on-feature))
            ?f2 <- (feature (number ?on-feature))
            (test (>= (length$ (fact-slot-value ?f1 tolerance)) 2))
            (operation (TAD left) (on-feature =(feature-with-tightest-tolerance ?f1)))
=>          (modify ?f1 (TAD left) (relation-with-feature =(update-relation-with-feature ?f1)) (tolerance =(update-tolerance ?f1)))
            (modify ?f2 (TAD left)))
```

```
(defrule MAIN::operation-on-feature-with-multiple-TAD-relation-with-one-feature-case2a
            ?f1 <- (operation (TAD right-left)(on-feature ?on-feature))
            ?f2 <- (feature (number ?on-feature))
            (test (= (length$ (fact-slot-value ?f1 tolerance)) 1))
            (operation (TAD right) (on-feature =(feature-with-tightest-tolerance-case-of-one-feature ?f1)))
=>          (modify ?f1 (TAD right) (relation-with-feature 0) (tolerance 0))
            (modify ?f2 (TAD right)))
(defrule MAIN::operation-on-feature-with-multiple-TAD-relation-with-multiple-features-case2
            ?f1 <- (operation (TAD right-left)(on-feature ?on-feature))
            ?f2 <- (feature (number ?on-feature))
            (test (>= (length$ (fact-slot-value ?f1 tolerance)) 2))
            (operation (TAD right) (on-feature =(feature-with-tightest-tolerance ?f1)))
=>          (modify ?f1 (TAD right) (relation-with-feature =(update-relation-with-feature ?f1)) (tolerance =(update-tolerance ?f1)))
            (modify ?f2 (TAD right)))
(defrule MAIN::operation-on-feature-with-multiple-TAD-relation-with-multiple-features-case4a
            ?f1 <- (operation (TAD right-left)(on-feature ?on-feature))
            ?f3 <- (feature (number ?on-feature))
            (test (>= (length$ (fact-slot-value ?f1 tolerance)) 2))
            ?f2 <- (operation (TAD right-left) (on-feature =(feature-with-tightest-tolerance ?f1)))
            (test (>= (length$ (fact-slot-value ?f2 tolerance)) 1))
            (operation (TAD left) (on-feature =(feature-with-tightest-tolerance-1 ?f1)))
=>          (modify ?f1 (TAD left) (relation-with-feature =(update-relation-with-feature-1 ?f1))(tolerance =(update-tolerance-1 ?f1)))
            (modify ?f3 (TAD left)))
(defrule MAIN::operation-on-feature-with-multiple-TAD-relation-with-multiple-features-case4
            ?f1 <- (operation (TAD right-left)(on-feature ?on-feature))
            ?f3 <- (feature (number ?on-feature))
            (test (>= (length$ (fact-slot-value ?f1 tolerance)) 2))
            ?f2 <- (operation (TAD right-left) (on-feature =(feature-with-tightest-tolerance ?f1)))
            (test (>= (length$ (fact-slot-value ?f2 tolerance)) 1))
            (operation (TAD right) (on-feature =(feature-with-tightest-tolerance-1 ?f1)))
=>          (modify ?f1 (TAD right) (relation-with-feature =(update-relation-with-feature-1 ?f1)) (tolerance =(update-tolerance-1
?f1)))
            (modify ?f3 (TAD right)))

;;;*****************************************************************
;;;* RULES FOR GENERATING PRECEDENCE CONSTRAINTS BETWEEN OPERATIONS*
;;;*****************************************************************

;;; The following rules are used to establish the precedence constraints between machining operations
;;; from the precedence constraints between features generated by the above rules

(deftemplate MAIN::operation_precedence
            (multislot values))
(defrule MAIN::operation_precedence_constraint_1
            (operation (number ?n1) (on-feature ?a) (machining_stage rough) )
            (operation (number ?n2) (on-feature ?a) (machining_stage rough) )
            (operation (number ?n3) (on-feature ?a) (machining_stage rough) )
            (feature_precedence ?a ?b)
            (operation (number ?n) (on-feature ?b) (machining_stage rough))
=>          (assert (operation_precedence (values ?n1 ?n)))
            (assert (operation_precedence (values ?n2 ?n)))
            (assert (operation_precedence (values ?n3 ?n))))
(deftemplate MAIN::operation_precedence2
            (multislot values))
(defrule MAIN::operation_precedence_constraint_2
            (operation (number ?n1) (on-feature ?a) (machining_stage semifinish) )
            (operation (number ?n2) (on-feature ?a) (machining_stage semifinish) )
            (operation (number ?n3) (on-feature ?a) (machining_stage semifinish) )
            (feature_precedence ?a ?b)
            (operation (number ?n) (on-feature ?b) (machining_stage semifinish))
=>          (assert (operation_precedence (values ?n1 ?n)))
            (assert (operation_precedence (values ?n2 ?n)))
            (assert (operation_precedence (values ?n3 ?n))))
(deftemplate MAIN::operation_precedence1
            (multislot values))
(defrule MAIN::operation_precedence_constraint_3
            (operation (number ?n1) (on-feature ?a) (machining_stage finish) )
            (operation (number ?n2) (on-feature ?a) (machining_stage finish) )
            (operation (number ?n3) (on-feature ?a) (machining_stage finish) )
            (feature_precedence ?a ?b)
            (operation (number ?n) (on-feature ?b) (machining_stage finish) )
=>          (assert (operation_precedence (values ?n1 ?n)))
            (assert (operation_precedence (values ?n2 ?n)))
            (assert (operation_precedence (values ?n3 ?n))))

;;;****************************************
;;;* RULES FOR REPORTING SETUP CLUSTERS  *
;;;****************************************
(deftemplate MAIN::setup-from-left-cluster
            (multislot operation_numbers
                        (type INTEGER)
                        (default ?NONE)))
(deftemplate MAIN::setup-from-right-cluster
            (multislot operation_numbers
                        (type INTEGER)
                        (default ?NONE)))
(defglobal ?*setup-from-left-cluster* = 0
              ?*setup-from-left* = 0
              ?*setup-from-right-cluster* = 0
              ?*setup-from-right* = 0)

;;; The following rules are used to report the cluster of operations to be performed in each set-up
;;; without regard to the sequence in which they are to be performed


(defrule MAIN::setup-from-left
            ?f1 <- (operation (TAD left))
=>          (bind ?*setup-from-left* (fact-slot-value ?f1 number))
            (bind ?*setup-from-left-cluster* (delete-member$ (create$ ?*setup-from-left-cluster* ?*setup-from-left*) 0)))
(defrule MAIN::setup-from-right
            ?f2 <- (operation (TAD right))
=>          (bind ?*setup-from-right* (fact-slot-value ?f2 number))
            (bind ?*setup-from-right-cluster* (delete-member$ (create$ ?*setup-from-right-cluster* ?*setup-from-right*) 0)))
(defrule report-setup-clusters
=>          (open "f:\\Sankha\\personal\\Research\\Actual\\clips_appl\\setup_planning_es\\pc_generation\\setup_l.clp" setup_l "w")
            (printout setup_l ?*setup-from-left-cluster*)
            (close setup_l)
            (printout t "The setup cluster of operations from the left are " ?*setup-from-left-cluster* crlf)
            (assert (setup-from-left-cluster (operation_numbers ?*setup-from-left-cluster*)))
            (open "f:\\Sankha\\personal\\Research\\Actual\\clips_appl\\setup_planning_es\\pc_generation\\setup_r.clp" setup_r "w")
            (printout setup_r ?*setup-from-right-cluster*)
            (close setup_r)
            (printout t "The setup cluster of operations from the right are " ?*setup-from-right-cluster* crlf)
            (assert (setup-from-right-cluster (operation_numbers ?*setup-from-right-cluster*)))
            (save-facts facts1.clp visible))

;;; Note that the results of execution of all the above rules are stored in the datafile, called facts1.clp
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; file name: program_file_2.CLP;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;==========================================================================================
;;;    Rules for determination of the preceding operations for each operation
;;;    CLIPS Version 6.10
;;;    To execute, merely load this program, then load the file,"facts1.clp", that is generated by "rules_set_1.clp", then reset and run.
;;;==========================================================================================
(defmodule MAIN
 (export deftemplate ?ALL))

;;;*****************************
;;;* DEFTEMPLATE DEFINITIONS *
;;;*****************************

(deftemplate MAIN::feature
            (slot number (type INTEGER) (default ?NONE))
            (slot name (type SYMBOL) (allowed-symbols CHAMFER EXTERNAL_STEP FACE GROOVE HOLE KEYWAY
EXTERNAL_TAPER THREAD HOLE))
            (slot type (type SYMBOL) (allowed-symbols EXTERNAL INTERNAL))
            (slot subtype (type SYMBOL) (allowed-symbols PRIMARY SECONDARY))
            (slot secondary_feature_to (type INTEGER) (default ?DERIVE))
            (multislot adjacent_features (type INTEGER) (default ?DERIVE))
            (multislot adjacent_features_names (type SYMBOL) (allowed-symbols CHAMFER EXTERNAL_STEP FACE
GROOVE HOLE KEYWAY EXTERNAL_TAPER THREAD HOLE))
            (multislot reference_features (type INTEGER) (default 0))
            (slot step_diameter (type NUMBER))
            (multislot adjacent_step_diameters (type NUMBER))
            (slot hole_diameter (type NUMBER))
            (slot hole_depth (type NUMBER))
            (multislot adjacent_hole_diameters (type NUMBER))
            (multislot adjacent_hole_depths (type NUMBER))
            (slot TAD (type SYMBOL) (allowed-symbols left right right-left) (default ?NONE)))
(deftemplate MAIN::operation
            (slot number (type INTEGER) (default ?NONE))
            (slot type (type SYMBOL) (default ?NONE))
            (slot machining_stage (type SYMBOL) (allowed-symbols rough semifinish finish) (default rough))
            (slot on-feature (type INTEGER) (default ?NONE))
            (slot TAD (type SYMBOL) (allowed-symbols left right right-left) (default ?NONE))
            (multislot relation-with-feature (type INTEGER) (default 0))
            (multislot tolerance (type NUMBER) (default 0)))
(deftemplate MAIN::setup-from-left-cluster (multislot operation_numbers))
(deftemplate MAIN::setup-from-right-cluster (multislot operation_numbers))
(deftemplate MAIN::operation_precedence (multislot values))

;;; The opn facts, which are infact modified operation facts, hold the information on each machining operation such as
;;; identification number, type, machining stage as well as the set-up cluster and the list of preceding operations, etc.

(deftemplate MAIN::opn
            (slot number (type INTEGER) (default ?NONE))
            (slot type (type SYMBOL))
            (slot machining_stage (type SYMBOL) (allowed-symbols rough semifinish finish) (default rough))
            (slot setup-cluster (type SYMBOL) (allowed-symbols left right))
            (multislot preceding_opn (type INTEGER) (default 0)))
(deftemplate MAIN::order-of-features-from-left-external-to-internal
            (multislot feature-numbers (type INTEGER) (default ?NONE)))
(deftemplate MAIN::order-of-features-from-right-external-to-internal
            (multislot feature-numbers (type INTEGER) (default ?NONE)))
(deftemplate MAIN::feature-with-largest-dia
            (slot number (type INTEGER) (default ?NONE)))

;;;******************************************************************************
;;;* RULES FOR DETERMINING THE PRECEDING OPERATIONS FOR EACH OPERATION*
;;;******************************************************************************

;;; The following rules are used to make a list of all the preceding operations for a given machining operation
;;; from the precedence constraints between operations generated earlier
```

```
(defrule MAIN::opn-1
            (operation (number ?n1) (TAD ?TAD) )
            (operation (number ?n2) (type ?type) (TAD ?TAD) (machining_stage ?stage))
            (operation_precedence (values ?n1 ?n2))
=>          (assert (opn (number ?n2) (type ?type) (machining_stage ?stage) (setup-cluster ?TAD) (preceding_opn ?n1))))
(defrule MAIN::opn-2
            ?f1 <- (opn (number ?n1) (type ?type) (setup-cluster ?TAD) (preceding_opn ?n2) (machining_stage ?stage))
            ?f2 <- (opn (number ?n1) (type ?type) (setup-cluster ?TAD) (preceding_opn ?n3) (machining_stage ?stage))
            (test (and (<> ?n2 ?n3) (<> ?n2 0) (<> ?n3 0)))
=>          (retract ?f1 ?f2)
            (assert (opn (number ?n1) (type ?type) (machining_stage ?stage) (setup-cluster ?TAD) (preceding_opn ?n2 ?n3))))
(defrule MAIN::opn-3
            ?f1 <- (opn (number ?n1) (type ?type) (setup-cluster ?TAD) (preceding_opn ?n2 ?n3) (machining_stage ?stage))
            ?f3 <- (opn (number ?n1) (type ?type) (setup-cluster ?TAD) (preceding_opn ?n4) (machining_stage ?stage))
            (test (and (<> ?n2 ?n3) (<> ?n2 ?n4) (<> ?n3 ?n4)(<> ?n2 0) (<> ?n3 0) (<> ?n4 0)))
=>          (retract ?f1 ?f3)
            (assert (opn (number ?n1) (type ?type) (machining_stage ?stage) (setup-cluster ?TAD) (preceding_opn ?n2 ?n3 ?n4))))
(defrule MAIN::opn-4
            (operation (number ?n) (type ?type) (TAD ?TAD) (machining_stage ?stage))
            (not (operation_precedence (values ? ?n)))
            (operation_precedence (values ?n ?))
=>          (assert (opn (number ?n) (type ?type) (machining_stage ?stage) (setup-cluster ?TAD) (preceding_opn 0))))
(defrule MAIN::opn-5
            ?f1 <- (opn (number ?n) (preceding_opn ?x&:(<> ?x 0)))
            ?f2 <- (opn (number ?n) (preceding_opn 0))
=>          (retract ?f2))
(defrule MAIN::opn-6
            (operation (number ?n) (type ?type) (machining_stage ?stage) (TAD ?TAD))
            (not (opn (number ?n)))
            (or (operation_precedence (values ?n ?))
               (not (operation_precedence (values ? ?n)))
               (not (operation_precedence (values ? ?n))))
=>          (assert (opn (number ?n) (type ?type) (machining_stage ?stage) (setup-cluster ?TAD) (preceding_opn 0))))
(defrule MAIN::opn-7
            ?f1 <- (opn (number ?n1) (type ?type) (machining_stage ?stage) (setup-cluster ?TAD) (preceding_opn ?n2 ?n3 ?n4))
            ?f3 <- (opn (number ?n1) (type ?type) (machining_stage ?stage) (setup-cluster ?TAD) (preceding_opn ?n5))
            (test (and (<> ?n2 ?n3) (<> ?n2 ?n4) (<> ?n3 ?n4) (<> ?n2 ?n5) (<> ?n3 ?n5) (<> ?n4 ?n5) (<> ?n2 0) (<> ?n3 0) (<> ?n4
0) (<> ?n5 0) ))
=>          (retract ?f1 ?f3)
            (assert (opn (number ?n1) (type ?type) (machining_stage ?stage) (setup-cluster ?TAD) (preceding_opn ?n2 ?n3 ?n4
?n5))))
(defrule MAIN::opn-8
            ?f1 <- (opn (number ?n1) (type ?type) (machining_stage ?stage) (setup-cluster ?TAD) (preceding_opn ?n2 ?n3 ?n4 ?n5))
            ?f3 <- (opn (number ?n1) (type ?type) (machining_stage ?stage) (setup-cluster ?TAD) (preceding_opn ?n6))
            (test (and (<> ?n2 ?n3) (<> ?n2 ?n4) (<> ?n3 ?n4) (<> ?n2 ?n5) (<> ?n3 ?n5) (<> ?n4 ?n5) (<> ?n2 ?n6) (<> ?n3 ?n6) (<>
?n4 ?n6) (<> ?n5 ?n6) (<> ?n2 0) (<> ?n3 0) (<> ?n4 0) (<> ?n5 0) (<> ?n6 0) ))
=>
            (retract ?f1 ?f3)
            (assert (opn (number ?n1) (type ?type) (machining_stage ?stage) (setup-cluster ?TAD) (preceding_opn ?n2 ?n3 ?n4 ?n5
?n6))))
(defrule MAIN::opn-9
            ?f1 <- (opn (number ?n1) (type ?type) (machining_stage ?stage) (setup-cluster ?TAD) (preceding_opn ?n2 ?n3 ?n4 ?n5
?n6))
            ?f3 <- (opn (number ?n1) (type ?type) (machining_stage ?stage) (setup-cluster ?TAD) (preceding_opn ?n7))
            (test (and (<> ?n2 ?n3) (<> ?n2 ?n4) (<> ?n3 ?n4) (<> ?n2 ?n5) (<> ?n3 ?n5) (<> ?n4 ?n5) (<> ?n2 ?n6) (<> ?n3 ?n6) (<>
?n4 ?n6) (<> ?n5 ?n6) (<> ?n2 ?n7) (<> ?n3 ?n7) (<> ?n4 ?n7) (<> ?n5 ?n7) (<> ?n6 ?n7) (<> ?n2 0) (<> ?n3 0) (<> ?n4 0) (<> ?n5
0) (<> ?n6 0) (<> ?n7 0) ))
=>          (retract ?f1 ?f3)
            (assert (opn (number ?n1) (type ?type) (machining_stage ?stage) (setup-cluster ?TAD) (preceding_opn ?n2 ?n3 ?n4 ?n5
?n6 ?n7))))
(defrule save_to_file
=>          (save-facts facts2.clp visible))

;;; Note that the results of execution of all the above rules are stored in the datafile, called facts2.clp
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; file name: program_file_3.CLP;;;;;;;;;;;;;;;;;
;;;===========================================================================================
;;;   Rules for determination of the sequence of operations
;;;   CLIPS Version 6.10
;;;
;;;   To execute, merely load this program, then load the file,"facts2.clp", that is generated by "rules_set_2.clp", then reset and run.
;;;===========================================================================================

(defmodule MAIN
 (export deftemplate ?ALL))

;;;***************************
;;;* DEFTEMPLATE DEFINITIONS *
;;;***************************

(deftemplate MAIN::feature
        (slot number (type INTEGER) (default ?NONE))
        (slot name (type SYMBOL) (allowed-symbols CHAMFER EXTERNAL_STEP FACE GROOVE HOLE KEYWAY
EXTERNAL_TAPER THREAD HOLE))
        (slot type (type SYMBOL) (allowed-symbols EXTERNAL INTERNAL))
        (slot subtype (type SYMBOL) (allowed-symbols PRIMARY SECONDARY))
        (multislot secondary_feature_to (type INTEGER) (default ?DERIVE))
        (multislot adjacent_features (type INTEGER) (default ?DERIVE))
        (multislot adjacent_features_names (type SYMBOL) (allowed-symbols CHAMFER EXTERNAL_STEP FACE
GROOVE HOLE KEYWAY EXTERNAL_TAPER THREAD HOLE))
        (multislot reference_features (type INTEGER) (default 0))
        (slot step_diameter (type NUMBER))
        (multislot adjacent_step_diameters (type NUMBER))
        (slot hole_diameter (type NUMBER))
        (slot hole_depth (type NUMBER))
        (multislot adjacent_hole_diameters (type NUMBER))
        (multislot adjacent_hole_depths (type NUMBER))
        (slot TAD (type SYMBOL) (allowed-symbols left right right-left) (default ?NONE)))
(deftemplate MAIN::operation
        (slot number (type INTEGER) (default ?NONE))
        (slot type (type SYMBOL) (default ?NONE))
        (slot machining_stage (type SYMBOL) (allowed-symbols rough semifinish finish))
        (slot on-feature (type INTEGER) (default ?NONE))
        (slot TAD (type SYMBOL) (allowed-symbols left right right-left) (default ?NONE))
        (multislot relation-with-feature (type INTEGER) (default 0))
        (multislot tolerance (type NUMBER) (default 0)))
(deftemplate MAIN::setup-from-left-cluster
        (multislot operation_numbers))
(deftemplate MAIN::setup-from-right-cluster
        (multislot operation_numbers))
(deftemplate MAIN::operation_precedence
        (multislot values))
(deftemplate MAIN::opn
        (slot number (type INTEGER) (default ?NONE))
        (slot type (type SYMBOL))
        (slot machining_stage (type SYMBOL) (allowed-symbols rough semifinish finish))
        (slot setup-cluster (type SYMBOL) (allowed-symbols left right))
        (multislot preceding_opn (type INTEGER) (default 0)))
(deftemplate MAIN::order-of-features-from-left-external-to-internal
        (multislot feature-numbers (type INTEGER) (default ?NONE)))
(deftemplate MAIN::order-of-features-from-right-external-to-internal
        (multislot feature-numbers (type INTEGER) (default ?NONE)))
(deftemplate MAIN::feature-with-largest-dia
        (slot number (type INTEGER) (default ?NONE)))

;;;******************************
;;;* RULES FOR GENERATING OPERATION SEQUENCE*
;;;******************************
(defglobal ?*sequence-left-cluster* = 0
           ?*opn-left-cluster* = 0 )
```

```
;;; The following rules are used to determine the sequence of operations in each set-up, subject to the precedence constraints
;;; generated earlier as well as various manufacturing logic for ordering the operations such as first machining of external
;;; surfaces (such as faces, steps, tapers, threads, grooves, etc.), followed by machining of internal surfaces (such as holes),
;;; and also first rough machining, followed by semi-finish machining, followed by finish machining. The above priority order
;;; for sequencing of machining operations has been implemented by using a feature in CLIPS, called salience.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; rules_set_3-left-rough.CLP;;;;;;;;;;;;;;;;;;;;;

;;;The following rules are used to determine the sequence of operations for the left set-up

(defrule MAIN::sequence-left-cluster-1
            (declare (salience 100))
            ?f1 <- (opn (number ?n)  (machining_stage rough) (setup-cluster left) (preceding_opn 0))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (eq  ?*sequence-left-cluster* 0)
             then
             (bind ?*sequence-left-cluster* (delete-member$ (create$ ?*sequence-left-cluster* ?*opn-left-cluster*) 0))
             else
             (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))

;;;;EXTERNAL_STEP;;;;;;
(defrule MAIN::sequence-left-cluster-2
            (declare (salience 99))
            ?f1 <- (opn (number ?n1) (machining_stage rough) (setup-cluster left) (preceding_opn ?n2))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_STEP))
            (test (not (= ?n2 0)))
            (opn (number ?n2)  (machining_stage rough) (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2) (create$ ?*sequence-left-cluster*))
               then
               (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-3
            (declare (salience 98))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster left) (preceding_opn ?n2 ?n3))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_STEP))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0))))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3) (create$ ?*sequence-left-cluster*))
               then (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-4
            (declare (salience 97))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_STEP))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0))))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
            (opn (number ?n4) (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4) (create$ ?*sequence-left-cluster*))
               then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-5
            (declare (salience 96))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4 ?n5))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_STEP))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) ))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
            (opn (number ?n4)  (setup-cluster left))
            (opn (number ?n5)  (setup-cluster left))
```

```
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5) (create$ ?*sequence-left-cluster*))
              then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-6
            (declare (salience 95))
            ?f1 <- (opn (number ?n1) (machining_stage rough) (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_STEP))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) ))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
            (opn (number ?n4)  (setup-cluster left))
            (opn (number ?n5)  (setup-cluster left))
            (opn (number ?n6)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6) (create$ ?*sequence-left-cluster*))
              then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-7
            (declare (salience 94))
            ?f1 <- (opn (number ?n1) (machining_stage rough) (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6 ?n7))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_STEP))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) (not (= ?n7 0)) ))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
            (opn (number ?n4)  (setup-cluster left))
            (opn (number ?n5)  (setup-cluster left))
            (opn (number ?n6)  (setup-cluster left))
            (opn (number ?n7)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6 ?n7) (create$ ?*sequence-left-cluster*))
              then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))

;;;;;FACE;;;;;
(defrule MAIN::sequence-left-cluster-2e
            (declare (salience 86))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster left) (preceding_opn ?n2))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name FACE))
            (test (not (= ?n2 0)))
            (opn (number ?n2)  (setup-cluster left))
=>
            (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2) (create$ ?*sequence-left-cluster*))
              then
              (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-3e
            (declare (salience 85))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster left) (preceding_opn ?n2 ?n3))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name FACE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0))))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3) (create$ ?*sequence-left-cluster*))
              then (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-4e
            (declare (salience 84))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name FACE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0))))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
            (opn (number ?n4)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4) (create$ ?*sequence-left-cluster*))
               then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-5e
            (declare (salience 83))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4 ?n5))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name FACE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) ))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
            (opn (number ?n4)  (setup-cluster left))
            (opn (number ?n5)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5) (create$ ?*sequence-left-cluster*))
               then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))

;;;;;;HOLE;;;;;;;
(defrule MAIN::sequence-left-cluster-2a
            (declare (salience 40))
            ?f1 <- (opn (number ?n1) (machining_stage rough) (setup-cluster left) (preceding_opn ?n2))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name HOLE))
            (test (not (= ?n2 0)))
            (opn (number ?n2) (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2) (create$ ?*sequence-left-cluster*))
               then
               (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-3a
            (declare (salience 39))
            ?f1 <- (opn (number ?n1) (machining_stage rough) (setup-cluster left) (preceding_opn ?n2 ?n3))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name HOLE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0))))
            (opn (number ?n2) (setup-cluster left))
            (opn (number ?n3) (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3) (create$ ?*sequence-left-cluster*))
               then (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-4a
            (declare (salience 38))
            ?f1 <- (opn (number ?n1) (machining_stage rough) (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name HOLE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0))))
            (opn (number ?n2) (setup-cluster left))
            (opn (number ?n3) (setup-cluster left))
            (opn (number ?n4) (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4) (create$ ?*sequence-left-cluster*))
               then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-5a
            (declare (salience 37))
            ?f1 <- (opn (number ?n1) (machining_stage rough) (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4 ?n5))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name HOLE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) ))
            (opn (number ?n2) (setup-cluster left))
            (opn (number ?n3) (setup-cluster left))
            (opn (number ?n4) (setup-cluster left))
            (opn (number ?n5) (setup-cluster left))
```

```
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5) (create$ ?*sequence-left-cluster*))
               then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-6a
            (declare (salience 36))
            ?f1 <- (opn (number ?n1) (machining_stage rough) (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name HOLE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) ))
            (opn (number ?n2) (setup-cluster left))
            (opn (number ?n3) (setup-cluster left))
            (opn (number ?n4) (setup-cluster left))
            (opn (number ?n5) (setup-cluster left))
            (opn (number ?n6) (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6) (create$ ?*sequence-left-cluster*))
               then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-7a
            (declare (salience 35))
            ?f1 <- (opn (number ?n1) (machining_stage rough) (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6 ?n7))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name HOLE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) (not (= ?n7 0)) ))
            (opn (number ?n2) (setup-cluster left))
            (opn (number ?n3) (setup-cluster left))
            (opn (number ?n4) (setup-cluster left))
            (opn (number ?n5) (setup-cluster left))
            (opn (number ?n6) (setup-cluster left))
            (opn (number ?n7) (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6 ?n7) (create$ ?*sequence-left-cluster*))
               then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))

;;;;GROOVE;;;;;
(defrule MAIN::sequence-left-cluster-2b
            (declare (salience 46))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster left) (preceding_opn ?n2))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name GROOVE))
            (test (not (= ?n2 0)))
            (opn (number ?n2)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2) (create$ ?*sequence-left-cluster*))
               then
               (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-3b
            (declare (salience 45))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster left) (preceding_opn ?n2 ?n3))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name GROOVE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0))))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3) (create$ ?*sequence-left-cluster*))
               then (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-4cg
            (declare (salience 44))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name GROOVE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0))))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
            (opn (number ?n4)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4) (create$ ?*sequence-left-cluster*))
              then (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-5cg
            (declare (salience 43))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4 ?n5))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name GROOVE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5 0))))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
            (opn (number ?n4)  (setup-cluster left))
            (opn (number ?n5)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5) (create$ ?*sequence-left-cluster*))
              then (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))

;;;;;;THREAD;;;;;
(defrule MAIN::sequence-left-cluster-2c
            (declare (salience 42))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster left) (preceding_opn ?n2))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name THREAD))
            (test (not (= ?n2 0)))
            (opn (number ?n2)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2) (create$ ?*sequence-left-cluster*))
              then
              (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-3c
            (declare (salience 41))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster left) (preceding_opn ?n2 ?n3))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name THREAD))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0))))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3) (create$ ?*sequence-left-cluster*))
              then (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-4c
            (declare (salience 40))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name THREAD))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0))))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
            (opn (number ?n4)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4) (create$ ?*sequence-left-cluster*))
              then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))

;;;;;;EXTERNAL_TAPER;;;;;;
(defrule MAIN::sequence-left-cluster-2d
            (declare (salience 93))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster left) (preceding_opn ?n2))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_TAPER))
            (test (not (= ?n2 0)))
            (opn (number ?n2)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2) (create$ ?*sequence-left-cluster*))
              then
              (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
```

```
(defrule MAIN::sequence-left-cluster-3d
          (declare (salience 92))
          ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster left) (preceding_opn ?n2 ?n3))
          (operation (number ?n1)  (on-feature ?N1))
          (feature (number ?N1) (name EXTERNAL_TAPER))
          (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0))))
          (opn (number ?n2)  (setup-cluster left))
          (opn (number ?n3)  (setup-cluster left))
=>        (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
          (if (subsetp (create$ ?n2 ?n3) (create$ ?*sequence-left-cluster*))
            then (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-4d
          (declare (salience 91))
          ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4))
          (operation (number ?n1)  (on-feature ?N1))
          (feature (number ?N1) (name EXTERNAL_TAPER))
          (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0))))
          (opn (number ?n2)  (setup-cluster left))
          (opn (number ?n3)  (setup-cluster left))
          (opn (number ?n4)  (setup-cluster left))
=>        (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
          (if (subsetp (create$ ?n2 ?n3 ?n4) (create$ ?*sequence-left-cluster*))
            then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-5d
          (declare (salience 90))
          ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4 ?n5))
          (operation (number ?n1)  (on-feature ?N1))
          (feature (number ?N1) (name EXTERNAL_TAPER))
          (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) ))
          (opn (number ?n2)  (setup-cluster left))
          (opn (number ?n3)  (setup-cluster left))
          (opn (number ?n4)  (setup-cluster left))
          (opn (number ?n5)  (setup-cluster left))
=>        (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
          (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5) (create$ ?*sequence-left-cluster*))
            then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-6d
          (declare (salience 89))
          ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6))
          (operation (number ?n1)  (on-feature ?N1))
          (feature (number ?N1) (name EXTERNAL_TAPER))
          (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) ))
          (opn (number ?n2)  (setup-cluster left))
          (opn (number ?n3)  (setup-cluster left))
          (opn (number ?n4)  (setup-cluster left))
          (opn (number ?n5)  (setup-cluster left))
          (opn (number ?n6)  (setup-cluster left))
=>        (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
          (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6) (create$ ?*sequence-left-cluster*))
            then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-7d
          (declare (salience 88))
          ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6 ?n7))
          (operation (number ?n1)  (on-feature ?N1))
          (feature (number ?N1) (name EXTERNAL_TAPER))
          (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) (not (= ?n7 0)) ))
          (opn (number ?n2)  (setup-cluster left))
          (opn (number ?n3)  (setup-cluster left))
          (opn (number ?n4)  (setup-cluster left))
          (opn (number ?n5)  (setup-cluster left))
          (opn (number ?n6)  (setup-cluster left))
          (opn (number ?n7)  (setup-cluster left))
=>        (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
```

```
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6 ?n7) (create$ ?*sequence-left-cluster*))
              then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))

;;;;KEYWAY;;;;
(defrule MAIN::sequence-left-cluster-2z
            (declare (salience 9))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster left) (preceding_opn ?n2))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name KEYWAY))
            (test (not (= ?n2 0)))
            (opn (number ?n2)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2) (create$ ?*sequence-left-cluster*))
              then
              (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-3z
            (declare (salience 8))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster left) (preceding_opn ?n2 ?n3))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name KEYWAY))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0))))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3) (create$ ?*sequence-left-cluster*))
              then (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-4z
            (declare (salience 7))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name KEYWAY))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0))))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
            (opn (number ?n4)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4) (create$ ?*sequence-left-cluster*))
              then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))


(defrule report-sequence-setup-clusters-left
=>          (open "f:\\Sankha\\personal\\Research\\Actual\\clips_appl\\setup_planning_es\\pc_generation\\sequence_l.clp" sequence_l
"w")
            (printout sequence_l ?*sequence-left-cluster*)
            (close sequence_l)
            (printout t "The sequence of operations from the left is " ?*sequence-left-cluster* crlf))
(defrule save_to_file
=>          (save-facts facts3.clp visible))



;;;;;;;;;;;;;;;;;;;;;;;;;;rules_set_3-left-semifinish.CLP;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule MAIN::sequence-left-cluster-semifinish-1
            (declare (salience 80))
            ?f1 <- (opn (number ?n)  (machining_stage semifinish) (setup-cluster left) (preceding_opn 0))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (eq  ?*sequence-left-cluster* 0)
             then
             (bind ?*sequence-left-cluster* (delete-member$ (create$ ?*sequence-left-cluster* ?*opn-left-cluster*) 0))
             else
             (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))

;;;;EXTERNAL_STEP;;;;;;
(defrule MAIN::sequence-left-cluster-semifinish-2
            (declare (salience 79))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish) (setup-cluster left) (preceding_opn ?n2))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_STEP))
            (test (not (= ?n2 0)))
            (opn (number ?n2)  (machining_stage semifinish) (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2) (create$ ?*sequence-left-cluster*))
               then
               (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-semifinish-3
            (declare (salience 78))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish)  (setup-cluster left) (preceding_opn ?n2 ?n3))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_STEP))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0))))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3) (create$ ?*sequence-left-cluster*))
               then (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-semifinish-4
            (declare (salience 77))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish)  (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_STEP))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0))))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
            (opn (number ?n4) (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4) (create$ ?*sequence-left-cluster*))
               then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-semifinish-5
            (declare (salience 76))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish)  (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4 ?n5))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_STEP))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) ))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
            (opn (number ?n4)  (setup-cluster left))
            (opn (number ?n5)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5) (create$ ?*sequence-left-cluster*))
               then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-semifinish-6
            (declare (salience 75))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish) (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_STEP))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) ))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
            (opn (number ?n4)  (setup-cluster left))
            (opn (number ?n5)  (setup-cluster left))
            (opn (number ?n6)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6) (create$ ?*sequence-left-cluster*))
               then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-semifinish-7
            (declare (salience 74))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish) (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6 ?n7))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_STEP))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) (not (= ?n7 0)) ))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
            (opn (number ?n4)  (setup-cluster left))
            (opn (number ?n5)  (setup-cluster left))
            (opn (number ?n6)  (setup-cluster left))
            (opn (number ?n7)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6 ?n7) (create$ ?*sequence-left-cluster*))
              then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))

;;;;;FACE;;;;;
(defrule MAIN::sequence-left-cluster-semifinish-2e
            (declare (salience 67))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish)  (setup-cluster left) (preceding_opn ?n2))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name FACE))
            (test (not (= ?n2 0)))
            (opn (number ?n2)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2) (create$ ?*sequence-left-cluster*))
              then
              (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-semifinish-3e
            (declare (salience 66))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish)  (setup-cluster left) (preceding_opn ?n2 ?n3))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name FACE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0))))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3) (create$ ?*sequence-left-cluster*))
              then (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-semifinish-4e
            (declare (salience 65))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish)  (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name FACE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0))))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
            (opn (number ?n4)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4) (create$ ?*sequence-left-cluster*))
              then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-semifinish-5e
            (declare (salience 64))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish)  (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4 ?n5))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name FACE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) ))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
            (opn (number ?n4)  (setup-cluster left))
            (opn (number ?n5)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5) (create$ ?*sequence-left-cluster*))
              then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))

;;;;;;HOLE;;;;;;
 (defrule MAIN::sequence-left-cluster-semifinish-2a
            (declare (salience 28))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish) (setup-cluster left) (preceding_opn ?n2))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name HOLE))
            (test (not (= ?n2 0)))
            (opn (number ?n2) (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2) (create$ ?*sequence-left-cluster*))
               then
               (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-semifinish-3a
            (declare (salience 27))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish) (setup-cluster left) (preceding_opn ?n2 ?n3))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name HOLE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0))))
            (opn (number ?n2) (setup-cluster left))
            (opn (number ?n3) (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3) (create$ ?*sequence-left-cluster*))
               then (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-semifinish-4a
            (declare (salience 26))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish) (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name HOLE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0))))
            (opn (number ?n2) (setup-cluster left))
            (opn (number ?n3) (setup-cluster left))
            (opn (number ?n4) (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4) (create$ ?*sequence-left-cluster*))
               then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-semifinish-5a
            (declare (salience 25))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish) (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4 ?n5))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name HOLE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) ))
            (opn (number ?n2) (setup-cluster left))
            (opn (number ?n3) (setup-cluster left))
            (opn (number ?n4) (setup-cluster left))
            (opn (number ?n5) (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5) (create$ ?*sequence-left-cluster*))
               then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-semifinish-6a
            (declare (salience 24))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish) (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name HOLE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) ))
            (opn (number ?n2) (setup-cluster left))
            (opn (number ?n3) (setup-cluster left))
            (opn (number ?n4) (setup-cluster left))
            (opn (number ?n5) (setup-cluster left))
            (opn (number ?n6) (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6) (create$ ?*sequence-left-cluster*))
               then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-semifinish-7a
            (declare (salience 23))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish) (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6 ?n7))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name HOLE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) (not (= ?n7 0)) ))
            (opn (number ?n2) (setup-cluster left))
            (opn (number ?n3) (setup-cluster left))
            (opn (number ?n4) (setup-cluster left))
            (opn (number ?n5) (setup-cluster left))
            (opn (number ?n6) (setup-cluster left))
            (opn (number ?n7) (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6 ?n7) (create$ ?*sequence-left-cluster*))
              then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))

;;;;;;EXTERNAL_TAPER;;;;;;
(defrule MAIN::sequence-left-cluster-semifinish-2d
            (declare (salience 73))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish)  (setup-cluster left) (preceding_opn ?n2))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_TAPER))
            (test (not (= ?n2 0)))
            (opn (number ?n2)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2) (create$ ?*sequence-left-cluster*))
              then
              (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-semifinish-3d
            (declare (salience 72))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish)  (setup-cluster left) (preceding_opn ?n2 ?n3))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_TAPER))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0))))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3) (create$ ?*sequence-left-cluster*))
              then (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-semifinish-4d
            (declare (salience 71))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish)  (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_TAPER))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0))))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
            (opn (number ?n4)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4) (create$ ?*sequence-left-cluster*))
              then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-semifinish-5d
            (declare (salience 70))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish)  (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4 ?n5))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_TAPER))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) ))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
            (opn (number ?n4)  (setup-cluster left))
            (opn (number ?n5)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5) (create$ ?*sequence-left-cluster*))
              then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-semifinish-6d
            (declare (salience 69))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish)  (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_TAPER))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) ))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
            (opn (number ?n4)  (setup-cluster left))
            (opn (number ?n5)  (setup-cluster left))
            (opn (number ?n6)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6) (create$ ?*sequence-left-cluster*))
              then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-semifinish-7d
            (declare (salience 68))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish)  (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6 ?n7))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_TAPER))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) (not (= ?n7 0)) ))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
            (opn (number ?n4)  (setup-cluster left))
            (opn (number ?n5)  (setup-cluster left))
            (opn (number ?n6)  (setup-cluster left))
            (opn (number ?n7)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6 ?n7) (create$ ?*sequence-left-cluster*))
              then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule report-sequence-setup-clusters-left
=>          (open "f:\\Sankha\\personal\\Research\\Actual\\clips_appl\\setup_planning_es\\pc_generation\\sequence_l.clp" sequence_l
"w")
            (printout sequence_l ?*sequence-left-cluster*)
            (close sequence_l)
            (printout t "The sequence of operations from the left is " ?*sequence-left-cluster* crlf))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; rules_set_3-left-finish.CLP;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule MAIN::sequence-left-cluster-finish-1
            (declare (salience 63))
            ?f1 <- (opn (number ?n)  (machining_stage finish) (setup-cluster left) (preceding_opn 0))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (eq  ?*sequence-left-cluster* 0)
             then
             (bind ?*sequence-left-cluster* (delete-member$ (create$ ?*sequence-left-cluster* ?*opn-left-cluster*) 0))
             else
             (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))

;;;;EXTERNAL_STEP;;;;;;
(defrule MAIN::sequence-left-cluster-finish-2
            (declare (salience 62))
            ?f1 <- (opn (number ?n1) (machining_stage finish) (setup-cluster left) (preceding_opn ?n2))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_STEP))
            (test (not (= ?n2 0)))
            (opn (number ?n2)  (machining_stage finish) (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2) (create$ ?*sequence-left-cluster*))
              then
              (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-finish-3
            (declare (salience 61))
            ?f1 <- (opn (number ?n1) (machining_stage finish)  (setup-cluster left) (preceding_opn ?n2 ?n3))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_STEP))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0))))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
```

```
            (if (subsetp (create$ ?n2 ?n3) (create$ ?*sequence-left-cluster*))
              then (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-finish-4
            (declare (salience 60))
            ?f1 <- (opn (number ?n1) (machining_stage finish)  (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_STEP))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0))))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
            (opn (number ?n4) (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4) (create$ ?*sequence-left-cluster*))
              then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-finish-5
            (declare (salience 59))
            ?f1 <- (opn (number ?n1) (machining_stage finish)  (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4 ?n5))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_STEP))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) ))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
            (opn (number ?n4)  (setup-cluster left))
            (opn (number ?n5)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5) (create$ ?*sequence-left-cluster*))
              then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-finish-6
            (declare (salience 58))
            ?f1 <- (opn (number ?n1) (machining_stage finish) (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_STEP))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) ))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
            (opn (number ?n4)  (setup-cluster left))
            (opn (number ?n5)  (setup-cluster left))
            (opn (number ?n6)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6) (create$ ?*sequence-left-cluster*))
              then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-finish-7
            (declare (salience 57))
            ?f1 <- (opn (number ?n1) (machining_stage finish) (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6 ?n7))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_STEP))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) (not (= ?n7 0)) ))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
            (opn (number ?n4)  (setup-cluster left))
            (opn (number ?n5)  (setup-cluster left))
            (opn (number ?n6)  (setup-cluster left))
            (opn (number ?n7)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6 ?n7) (create$ ?*sequence-left-cluster*))
              then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))

;;;;;FACE;;;;;
(defrule MAIN::sequence-left-cluster-finish-2e
            (declare (salience 50))
            ?f1 <- (opn (number ?n1) (machining_stage finish)  (setup-cluster left) (preceding_opn ?n2))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name FACE))
            (test (not (= ?n2 0)))
            (opn (number ?n2)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2) (create$ ?*sequence-left-cluster*))
               then
               (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-finish-3e
            (declare (salience 49))
            ?f1 <- (opn (number ?n1) (machining_stage finish)  (setup-cluster left) (preceding_opn ?n2 ?n3))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name FACE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0))))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3) (create$ ?*sequence-left-cluster*))
               then (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-finish-4e
            (declare (salience 48))
            ?f1 <- (opn (number ?n1) (machining_stage finish)  (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name FACE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0))))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
            (opn (number ?n4)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4) (create$ ?*sequence-left-cluster*))
               then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-finish-5e
            (declare (salience 47))
            ?f1 <- (opn (number ?n1) (machining_stage finish)  (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4 ?n5))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name FACE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) ))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
            (opn (number ?n4)  (setup-cluster left))
            (opn (number ?n5)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5) (create$ ?*sequence-left-cluster*))
               then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))

;;;;;;HOLE;;;;;;
(defrule MAIN::sequence-left-cluster-finish-2a
            (declare (salience 22))
            ?f1 <- (opn (number ?n1) (machining_stage finish) (setup-cluster left) (preceding_opn ?n2))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name HOLE))
            (test (not (= ?n2 0)))
            (opn (number ?n2) (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2) (create$ ?*sequence-left-cluster*))
               then
               (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-finish-3a
            (declare (salience 21))
            ?f1 <- (opn (number ?n1) (machining_stage finish) (setup-cluster left) (preceding_opn ?n2 ?n3))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name HOLE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0))))
            (opn (number ?n2) (setup-cluster left))
            (opn (number ?n3) (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3) (create$ ?*sequence-left-cluster*))
            then (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-finish-4a
         (declare (salience 20))
         ?f1 <- (opn (number ?n1) (machining_stage finish) (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4))
         (operation (number ?n1) (on-feature ?N1))
         (feature (number ?N1) (name HOLE))
         (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0))))
         (opn (number ?n2) (setup-cluster left))
         (opn (number ?n3) (setup-cluster left))
         (opn (number ?n4) (setup-cluster left))
=>       (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
         (if (subsetp (create$ ?n2 ?n3 ?n4) (create$ ?*sequence-left-cluster*))
            then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-finish-5a
         (declare (salience 19))
         ?f1 <- (opn (number ?n1) (machining_stage finish) (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4 ?n5))
         (operation (number ?n1) (on-feature ?N1))
         (feature (number ?N1) (name HOLE))
         (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) ))
         (opn (number ?n2) (setup-cluster left))
         (opn (number ?n3) (setup-cluster left))
         (opn (number ?n4) (setup-cluster left))
         (opn (number ?n5) (setup-cluster left))
=>       (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
         (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5) (create$ ?*sequence-left-cluster*))
            then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-finish-6a
         (declare (salience 18))
         ?f1 <- (opn (number ?n1) (machining_stage finish) (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6))
         (operation (number ?n1) (on-feature ?N1))
         (feature (number ?N1) (name HOLE))
         (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) ))
         (opn (number ?n2) (setup-cluster left))
         (opn (number ?n3) (setup-cluster left))
         (opn (number ?n4) (setup-cluster left))
         (opn (number ?n5) (setup-cluster left))
         (opn (number ?n6) (setup-cluster left))
=>       (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
         (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6) (create$ ?*sequence-left-cluster*))
            then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-finish-7a
         (declare (salience 17))
         ?f1 <- (opn (number ?n1) (machining_stage finish) (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6 ?n7))
         (operation (number ?n1) (on-feature ?N1))
         (feature (number ?N1) (name HOLE))
         (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) (not (= ?n7 0)) ))
         (opn (number ?n2) (setup-cluster left))
         (opn (number ?n3) (setup-cluster left))
         (opn (number ?n4) (setup-cluster left))
         (opn (number ?n5) (setup-cluster left))
         (opn (number ?n6) (setup-cluster left))
         (opn (number ?n7) (setup-cluster left))
=>       (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
         (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6 ?n7) (create$ ?*sequence-left-cluster*))
            then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))

;;;;;;EXTERNAL_TAPER;;;;;;
(defrule MAIN::sequence-left-cluster-finish-2d
         (declare (salience 56))
         ?f1 <- (opn (number ?n1) (machining_stage finish)  (setup-cluster left) (preceding_opn ?n2))
         (operation (number ?n1)  (on-feature ?N1))
         (feature (number ?N1) (name EXTERNAL_TAPER))
         (test (not (= ?n2 0)))
```

```
            (opn (number ?n2)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2) (create$ ?*sequence-left-cluster*))
               then
               (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-finish-3d
            (declare (salience 55))
            ?f1 <- (opn (number ?n1) (machining_stage finish)  (setup-cluster left) (preceding_opn ?n2 ?n3))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_TAPER))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0))))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3) (create$ ?*sequence-left-cluster*))
               then (bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-finish-4d
            (declare (salience 54))
            ?f1 <- (opn (number ?n1) (machining_stage finish)  (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_TAPER))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0))))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
            (opn (number ?n4)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4) (create$ ?*sequence-left-cluster*))
               then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-finish-5d
            (declare (salience 53))
            ?f1 <- (opn (number ?n1) (machining_stage finish)  (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4 ?n5))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_TAPER))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) ))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
            (opn (number ?n4)  (setup-cluster left))
            (opn (number ?n5)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5) (create$ ?*sequence-left-cluster*))
               then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-finish-6d
            (declare (salience 52))
            ?f1 <- (opn (number ?n1) (machining_stage finish)  (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_TAPER))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) ))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
            (opn (number ?n4)  (setup-cluster left))
            (opn (number ?n5)  (setup-cluster left))
            (opn (number ?n6)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6) (create$ ?*sequence-left-cluster*))
               then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule MAIN::sequence-left-cluster-finish-7d
            (declare (salience 51))
            ?f1 <- (opn (number ?n1) (machining_stage finish)  (setup-cluster left) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6 ?n7))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_TAPER))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) (not (= ?n7 0)) ))
            (opn (number ?n2)  (setup-cluster left))
            (opn (number ?n3)  (setup-cluster left))
            (opn (number ?n4)  (setup-cluster left))
            (opn (number ?n5)  (setup-cluster left))
            (opn (number ?n6)  (setup-cluster left))
            (opn (number ?n7)  (setup-cluster left))
=>          (bind ?*opn-left-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6 ?n7) (create$ ?*sequence-left-cluster*))
               then(bind ?*sequence-left-cluster* (create$ ?*sequence-left-cluster* ?*opn-left-cluster*))))
(defrule report-sequence-setup-clusters-left
=>          (open "f:\\Sankha\\personal\\Research\\Actual\\clips_appl\\setup_planning_es\\pc_generation\\sequence_l.clp" sequence_l
"w")
            (printout sequence_l ?*sequence-left-cluster*)
            (close sequence_l)
            (printout t "The sequence of operations from the left is " ?*sequence-left-cluster* crlf))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; rules_set_3-right-rough.CLP;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;The following rules are used to determine the sequence of operations for the right set-up

;;;******************************
;;;* RULES FOR GENERATING OPERATION SEQUENCE*
;;;******************************
(defglobal ?*sequence-right-cluster* = 0
             ?*opn-right-cluster* = 0)
(defrule MAIN::sequence-right-cluster-1
            (declare (salience 100))
            ?f1 <- (opn (number ?n)  (machining_stage rough) (setup-cluster right) (preceding_opn 0))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (eq  ?*sequence-right-cluster* 0)
              then
              (bind ?*sequence-right-cluster* (delete-member$ (create$ ?*sequence-right-cluster* ?*opn-right-cluster*) 0))
              else
              (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))

;;;;EXTERNAL_STEP;;;;;;
(defrule MAIN::sequence-right-cluster-2
            (declare (salience 99))
            ?f1 <- (opn (number ?n1) (machining_stage rough) (setup-cluster right) (preceding_opn ?n2))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_STEP))
            (test (not (= ?n2 0)))
            (opn (number ?n2)  (machining_stage rough) (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2) (create$ ?*sequence-right-cluster*))
               then
               (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-3
            (declare (salience 98))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster right) (preceding_opn ?n2 ?n3))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_STEP))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0))))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3) (create$ ?*sequence-right-cluster*))
               then (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-4
            (declare (salience 97))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_STEP))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0))))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
            (opn (number ?n4) (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
```

```
            (if (subsetp (create$ ?n2 ?n3 ?n4) (create$ ?*sequence-right-cluster*))
              then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-5
            (declare (salience 96))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4 ?n5))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_STEP))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) ))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
            (opn (number ?n4)  (setup-cluster right))
            (opn (number ?n5)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5) (create$ ?*sequence-right-cluster*))
              then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-6
            (declare (salience 95))
            ?f1 <- (opn (number ?n1) (machining_stage rough) (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_STEP))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) ))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
            (opn (number ?n4)  (setup-cluster right))
            (opn (number ?n5)  (setup-cluster right))
            (opn (number ?n6)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6) (create$ ?*sequence-right-cluster*))
              then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-7
            (declare (salience 94))
            ?f1 <- (opn (number ?n1) (machining_stage rough) (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6 ?n7))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_STEP))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) (not (= ?n7 0)) ))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
            (opn (number ?n4)  (setup-cluster right))
            (opn (number ?n5)  (setup-cluster right))
            (opn (number ?n6)  (setup-cluster right))
            (opn (number ?n7)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6 ?n7) (create$ ?*sequence-right-cluster*))
              then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))

;;;;;FACE;;;;;
(defrule MAIN::sequence-right-cluster-2e
            (declare (salience 86))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster right) (preceding_opn ?n2))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name FACE))
            (test (not (= ?n2 0)))
            (opn (number ?n2)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2) (create$ ?*sequence-right-cluster*))
              then
              (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-3e
            (declare (salience 85))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster right) (preceding_opn ?n2 ?n3))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name FACE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0))))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3) (create$ ?*sequence-right-cluster*))
               then (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-4e
            (declare (salience 84))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name FACE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0))))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
            (opn (number ?n4)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4) (create$ ?*sequence-right-cluster*))
               then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-5e
            (declare (salience 83))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4 ?n5))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name FACE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) ))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
            (opn (number ?n4)  (setup-cluster right))
            (opn (number ?n5)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5) (create$ ?*sequence-right-cluster*))
               then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))

;;;;;;HOLE;;;;;;;
(defrule MAIN::sequence-right-cluster-2a
            (declare (salience 40))
            ?f1 <- (opn (number ?n1) (machining_stage rough) (setup-cluster right) (preceding_opn ?n2))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name HOLE))
            (test (not (= ?n2 0)))
            (opn (number ?n2) (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2) (create$ ?*sequence-right-cluster*))
               then
               (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-3a
            (declare (salience 39))
            ?f1 <- (opn (number ?n1) (machining_stage rough) (setup-cluster right) (preceding_opn ?n2 ?n3))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name HOLE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0))))
            (opn (number ?n2) (setup-cluster right))
            (opn (number ?n3) (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3) (create$ ?*sequence-right-cluster*))
               then (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-4a
            (declare (salience 38))
            ?f1 <- (opn (number ?n1) (machining_stage rough) (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name HOLE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0))))
            (opn (number ?n2) (setup-cluster right))
            (opn (number ?n3) (setup-cluster right))
            (opn (number ?n4) (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4) (create$ ?*sequence-right-cluster*))
            then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-5a
        (declare (salience 37))
        ?f1 <- (opn (number ?n1) (machining_stage rough) (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4 ?n5))
        (operation (number ?n1) (on-feature ?N1))
        (feature (number ?N1) (name HOLE))
        (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) ))
        (opn (number ?n2) (setup-cluster right))
        (opn (number ?n3) (setup-cluster right))
        (opn (number ?n4) (setup-cluster right))
        (opn (number ?n5) (setup-cluster right))
=>      (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
        (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5) (create$ ?*sequence-right-cluster*))
            then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-6a
        (declare (salience 36))
        ?f1 <- (opn (number ?n1) (machining_stage rough) (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6))
        (operation (number ?n1) (on-feature ?N1))
        (feature (number ?N1) (name HOLE))
        (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) ))
        (opn (number ?n2) (setup-cluster right))
        (opn (number ?n3) (setup-cluster right))
        (opn (number ?n4) (setup-cluster right))
        (opn (number ?n5) (setup-cluster right))
        (opn (number ?n6) (setup-cluster right))
=>      (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
        (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6) (create$ ?*sequence-right-cluster*))
            then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-7a
        (declare (salience 35))
        ?f1 <- (opn (number ?n1) (machining_stage rough) (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6 ?n7))
        (operation (number ?n1) (on-feature ?N1))
        (feature (number ?N1) (name HOLE))
        (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) (not (= ?n7 0)) ))
        (opn (number ?n2) (setup-cluster right))
        (opn (number ?n3) (setup-cluster right))
        (opn (number ?n4) (setup-cluster right))
        (opn (number ?n5) (setup-cluster right))
        (opn (number ?n6) (setup-cluster right))
        (opn (number ?n7) (setup-cluster right))
=>      (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
        (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6 ?n7) (create$ ?*sequence-right-cluster*))
            then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))

;;;;GROOVE;;;;;
(defrule MAIN::sequence-right-cluster-2b
        (declare (salience 46))
        ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster right) (preceding_opn ?n2))
        (operation (number ?n1)  (on-feature ?N1))
        (feature (number ?N1) (name GROOVE))
        (test (not (= ?n2 0)))
        (opn (number ?n2)  (setup-cluster right))
=>      (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
        (if (subsetp (create$ ?n2) (create$ ?*sequence-right-cluster*))
            then
            (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-3b
        (declare (salience 45))
        ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster right) (preceding_opn ?n2 ?n3))
        (operation (number ?n1)  (on-feature ?N1))
        (feature (number ?N1) (name GROOVE))
        (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0))))
        (opn (number ?n2)  (setup-cluster right))
```

```
            (opn (number ?n3)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3) (create$ ?*sequence-right-cluster*))
              then (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-4cg
(declare (salience 44))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name GROOVE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0))))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
            (opn (number ?n4)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4) (create$ ?*sequence-right-cluster*))
              then (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-5cg
            (declare (salience 43))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4 ?n5))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name GROOVE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5 0))))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
            (opn (number ?n4)  (setup-cluster right))
            (opn (number ?n5)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5) (create$ ?*sequence-right-cluster*))
              then (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))

;;;;;;THREAD;;;;;
(defrule MAIN::sequence-right-cluster-2c
            (declare (salience 42))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster right) (preceding_opn ?n2))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name THREAD))
            (test (not (= ?n2 0)))
            (opn (number ?n2)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2) (create$ ?*sequence-right-cluster*))
              then
              (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-3c
            (declare (salience 41))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster right) (preceding_opn ?n2 ?n3))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name THREAD))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0))))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3) (create$ ?*sequence-right-cluster*))
              then (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-4c
            (declare (salience 40))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name THREAD))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0))))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
            (opn (number ?n4)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4) (create$ ?*sequence-right-cluster*))
              then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
```

```
;;;;;;EXTERNAL_TAPER;;;;;;

(defrule MAIN::sequence-right-cluster-2d
            (declare (salience 93))
            ?f1 <- (opn (number ?n1) (machining_stage rough) (setup-cluster right) (preceding_opn ?n2))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_TAPER))
            (test (not (= ?n2 0)))
            (opn (number ?n2) (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2) (create$ ?*sequence-right-cluster*))
               then
               (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-3d
            (declare (salience 92))
            ?f1 <- (opn (number ?n1) (machining_stage rough) (setup-cluster right) (preceding_opn ?n2 ?n3))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_TAPER))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0))))
            (opn (number ?n2) (setup-cluster right))
            (opn (number ?n3) (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3) (create$ ?*sequence-right-cluster*))
               then (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-4d
            (declare (salience 91))
            ?f1 <- (opn (number ?n1) (machining_stage rough) (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_TAPER))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0))))
            (opn (number ?n2) (setup-cluster right))
            (opn (number ?n3) (setup-cluster right))
            (opn (number ?n4) (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4) (create$ ?*sequence-right-cluster*))
               then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-5d
            (declare (salience 90))
            ?f1 <- (opn (number ?n1) (machining_stage rough) (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4 ?n5))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_TAPER))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) ))
            (opn (number ?n2) (setup-cluster right))
            (opn (number ?n3) (setup-cluster right))
            (opn (number ?n4) (setup-cluster right))
            (opn (number ?n5) (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5) (create$ ?*sequence-right-cluster*))
               then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-6d
            (declare (salience 89))
            ?f1 <- (opn (number ?n1) (machining_stage rough) (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_TAPER))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) ))
            (opn (number ?n2) (setup-cluster right))
            (opn (number ?n3) (setup-cluster right))
            (opn (number ?n4) (setup-cluster right))
            (opn (number ?n5) (setup-cluster right))
            (opn (number ?n6) (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6) (create$ ?*sequence-right-cluster*))
               then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-7d
            (declare (salience 88))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6 ?n7))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_TAPER))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) (not (= ?n7 0)) ))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
            (opn (number ?n4)  (setup-cluster right))
            (opn (number ?n5)  (setup-cluster right))
            (opn (number ?n6)  (setup-cluster right))
            (opn (number ?n7)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6 ?n7) (create$ ?*sequence-right-cluster*))
               then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))

;;;;;;KEYWAY;;;;;
(defrule MAIN::sequence-right-cluster-2z
            (declare (salience 9))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster right) (preceding_opn ?n2))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name KEYWAY))
            (test (not (= ?n2 0)))
            (opn (number ?n2)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2) (create$ ?*sequence-right-cluster*))
               then
               (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-3z
            (declare (salience 8))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster right) (preceding_opn ?n2 ?n3))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name KEYWAY))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0))))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3) (create$ ?*sequence-right-cluster*))
               then (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-4z
            (declare (salience 7))
            ?f1 <- (opn (number ?n1) (machining_stage rough)  (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name KEYWAY))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0))))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
            (opn (number ?n4)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4) (create$ ?*sequence-right-cluster*))
               then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))

(defrule report-sequence-setup-clusters-right
=>          (open "f:\\Sankha\\personal\\Research\\Actual\\clips_appl\\setup_planning_es\\pc_generation\\sequence_l.clp" sequence_l
"w")
            (printout sequence_l ?*sequence-right-cluster*)
            (close sequence_l)
            (printout t "The sequence of operations from the right is " ?*sequence-right-cluster* crlf))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; rules_set_3-right-semifinish.CLP;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule MAIN::sequence-right-cluster-semifinish-1
            (declare (salience 80))
            ?f1 <- (opn (number ?n)  (machining_stage semifinish) (setup-cluster right) (preceding_opn 0))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (eq  ?*sequence-right-cluster* 0)
                then
                (bind ?*sequence-right-cluster* (delete-member$ (create$ ?*sequence-right-cluster* ?*opn-right-cluster*) 0))
                else
                (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))

;;;;EXTERNAL_STEP;;;;;;
(defrule MAIN::sequence-right-cluster-semifinish-2
            (declare (salience 79))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish) (setup-cluster right) (preceding_opn ?n2))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_STEP))
            (test (not (= ?n2 0)))
            (opn (number ?n2)  (machining_stage semifinish) (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2) (create$ ?*sequence-right-cluster*))
              then
              (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-semifinish-3
            (declare (salience 78))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish)  (setup-cluster right) (preceding_opn ?n2 ?n3))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_STEP))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0))))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3) (create$ ?*sequence-right-cluster*))
              then (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-semifinish-4
            (declare (salience 77))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish)  (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_STEP))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0))))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
            (opn (number ?n4) (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4) (create$ ?*sequence-right-cluster*))
              then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-semifinish-5
            (declare (salience 76))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish)  (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4 ?n5))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_STEP))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) ))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
            (opn (number ?n4)  (setup-cluster right))
            (opn (number ?n5)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5) (create$ ?*sequence-right-cluster*))
              then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-semifinish-6
            (declare (salience 75))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish) (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_STEP))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) ))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
            (opn (number ?n4)  (setup-cluster right))
            (opn (number ?n5)  (setup-cluster right))
            (opn (number ?n6)  (setup-cluster right))
```

```
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6) (create$ ?*sequence-right-cluster*))
              then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-semifinish-7
            (declare (salience 74))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish) (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6 ?n7))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_STEP))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) (not (= ?n7 0)) ))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
            (opn (number ?n4)  (setup-cluster right))
            (opn (number ?n5)  (setup-cluster right))
            (opn (number ?n6)  (setup-cluster right))
            (opn (number ?n7)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6 ?n7) (create$ ?*sequence-right-cluster*))
              then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))

;;;;;FACE;;;;;
(defrule MAIN::sequence-right-cluster-semifinish-2e
            (declare (salience 67))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish)  (setup-cluster right) (preceding_opn ?n2))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name FACE))
            (test (not (= ?n2 0)))
            (opn (number ?n2)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2) (create$ ?*sequence-right-cluster*))
              then
              (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-semifinish-3e
            (declare (salience 66))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish)  (setup-cluster right) (preceding_opn ?n2 ?n3))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name FACE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0))))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3) (create$ ?*sequence-right-cluster*))
              then (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-semifinish-4e
            (declare (salience 65))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish)  (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name FACE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0))))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
            (opn (number ?n4)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4) (create$ ?*sequence-right-cluster*))
              then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-semifinish-5e
            (declare (salience 64))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish)  (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4 ?n5))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name FACE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) ))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
            (opn (number ?n4)  (setup-cluster right))
            (opn (number ?n5)  (setup-cluster right))
```

```
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5) (create$ ?*sequence-right-cluster*))
               then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))

;;;;;HOLE;;;;;;
(defrule MAIN::sequence-right-cluster-semifinish-2a
            (declare (salience 28))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish) (setup-cluster right) (preceding_opn ?n2))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name HOLE))
            (test (not (= ?n2 0)))
            (opn (number ?n2) (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2) (create$ ?*sequence-right-cluster*))
               then
               (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-semifinish-3a
            (declare (salience 27))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish) (setup-cluster right) (preceding_opn ?n2 ?n3))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name HOLE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0))))
            (opn (number ?n2) (setup-cluster right))
            (opn (number ?n3) (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3) (create$ ?*sequence-right-cluster*))
               then (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-semifinish-4a
            (declare (salience 26))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish) (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name HOLE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0))))
            (opn (number ?n2) (setup-cluster right))
            (opn (number ?n3) (setup-cluster right))
            (opn (number ?n4) (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4) (create$ ?*sequence-right-cluster*))
               then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-semifinish-5a
            (declare (salience 25))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish) (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4 ?n5))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name HOLE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) ))
            (opn (number ?n2) (setup-cluster right))
            (opn (number ?n3) (setup-cluster right))
            (opn (number ?n4) (setup-cluster right))
            (opn (number ?n5) (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5) (create$ ?*sequence-right-cluster*))
               then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-semifinish-6a
            (declare (salience 24))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish) (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name HOLE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) ))
            (opn (number ?n2) (setup-cluster right))
            (opn (number ?n3) (setup-cluster right))
            (opn (number ?n4) (setup-cluster right))
            (opn (number ?n5) (setup-cluster right))
            (opn (number ?n6) (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
```

```
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6) (create$ ?*sequence-right-cluster*))
               then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-semifinish-7a
            (declare (salience 23))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish) (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6 ?n7))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name HOLE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) (not (= ?n7 0)) ))
            (opn (number ?n2) (setup-cluster right))
            (opn (number ?n3) (setup-cluster right))
            (opn (number ?n4) (setup-cluster right))
            (opn (number ?n5) (setup-cluster right))
            (opn (number ?n6) (setup-cluster right))
            (opn (number ?n7) (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6 ?n7) (create$ ?*sequence-right-cluster*))
               then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))

;;;;;;EXTERNAL_TAPER;;;;;;
(defrule MAIN::sequence-right-cluster-semifinish-2d
            (declare (salience 71))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish)  (setup-cluster right) (preceding_opn ?n2))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_TAPER))
            (test (not (= ?n2 0)))
            (opn (number ?n2)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2) (create$ ?*sequence-right-cluster*))
               then
               (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-semifinish-3d
            (declare (salience 72))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish)  (setup-cluster right) (preceding_opn ?n2 ?n3))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_TAPER))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0))))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3) (create$ ?*sequence-right-cluster*))
               then (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-semifinish-4d
            (declare (salience 71))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish)  (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_TAPER))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0))))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
            (opn (number ?n4)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4) (create$ ?*sequence-right-cluster*))
               then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-semifinish-5d
            (declare (salience 70))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish)  (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4 ?n5))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_TAPER))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) ))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
            (opn (number ?n4)  (setup-cluster right))
            (opn (number ?n5)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
```

```
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5) (create$ ?*sequence-right-cluster*))
               then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-semifinish-6d
            (declare (salience 69))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish)  (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_TAPER))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) ))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
            (opn (number ?n4)  (setup-cluster right))
            (opn (number ?n5)  (setup-cluster right))
            (opn (number ?n6)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6) (create$ ?*sequence-right-cluster*))
               then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-semifinish-7d
            (declare (salience 68))
            ?f1 <- (opn (number ?n1) (machining_stage semifinish)  (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6 ?n7))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_TAPER))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) (not (= ?n7 0)) ))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
            (opn (number ?n4)  (setup-cluster right))
            (opn (number ?n5)  (setup-cluster right))
            (opn (number ?n6)  (setup-cluster right))
            (opn (number ?n7)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6 ?n7) (create$ ?*sequence-right-cluster*))
               then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule report-sequence-setup-clusters-right
=>          (open "f:\\Sankha\\personal\\Research\\Actual\\clips_appl\\setup_planning_es\\pc_generation\\sequence_l.clp" sequence_l
"w")
            (printout sequence_l ?*sequence-right-cluster*)
            (close sequence_l)
            (printout t "The sequence of operations from the right is " ?*sequence-right-cluster* crlf))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; rules_set_3-right-finish.CLP;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule MAIN::sequence-right-cluster-finish-1
            (declare (salience 63))
            ?f1 <- (opn (number ?n)  (machining_stage finish) (setup-cluster right) (preceding_opn 0))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (eq  ?*sequence-right-cluster* 0)
             then
             (bind ?*sequence-right-cluster* (delete-member$ (create$ ?*sequence-right-cluster* ?*opn-right-cluster*) 0))
             else
             (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))

;;;;EXTERNAL_STEP;;;;;;
(defrule MAIN::sequence-right-cluster-finish-2
            (declare (salience 62))
            ?f1 <- (opn (number ?n1) (machining_stage finish) (setup-cluster right) (preceding_opn ?n2))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_STEP))
            (test (not (= ?n2 0)))
            (opn (number ?n2)  (machining_stage finish) (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2) (create$ ?*sequence-right-cluster*))
               then
               (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-finish-3
            (declare (salience 61))
```

```
          ?f1 <- (opn (number ?n1) (machining_stage finish)  (setup-cluster right) (preceding_opn ?n2 ?n3))
          (operation (number ?n1)  (on-feature ?N1))
          (feature (number ?N1) (name EXTERNAL_STEP))
          (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0))))
          (opn (number ?n2)  (setup-cluster right))
          (opn (number ?n3)  (setup-cluster right))
=>        (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
          (if (subsetp (create$ ?n2 ?n3) (create$ ?*sequence-right-cluster*))
            then (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-finish-4
          (declare (salience 60))
          ?f1 <- (opn (number ?n1) (machining_stage finish)  (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4))
          (operation (number ?n1)  (on-feature ?N1))
          (feature (number ?N1) (name EXTERNAL_STEP))
          (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0))))
          (opn (number ?n2)  (setup-cluster right))
          (opn (number ?n3)  (setup-cluster right))
          (opn (number ?n4) (setup-cluster right))
=>        (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
          (if (subsetp (create$ ?n2 ?n3 ?n4) (create$ ?*sequence-right-cluster*))
            then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-finish-5
          (declare (salience 59))
          ?f1 <- (opn (number ?n1) (machining_stage finish)  (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4 ?n5))
          (operation (number ?n1)  (on-feature ?N1))
          (feature (number ?N1) (name EXTERNAL_STEP))
          (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) ))
          (opn (number ?n2)  (setup-cluster right))
          (opn (number ?n3)  (setup-cluster right))
          (opn (number ?n4)  (setup-cluster right))
          (opn (number ?n5)  (setup-cluster right))
=>        (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
          (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5) (create$ ?*sequence-right-cluster*))
            then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-finish-6
          (declare (salience 58))
          ?f1 <- (opn (number ?n1) (machining_stage finish) (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6))
          (operation (number ?n1) (on-feature ?N1))
          (feature (number ?N1) (name EXTERNAL_STEP))
          (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) ))
          (opn (number ?n2)  (setup-cluster right))
          (opn (number ?n3)  (setup-cluster right))
          (opn (number ?n4)  (setup-cluster right))
          (opn (number ?n5)  (setup-cluster right))
          (opn (number ?n6)  (setup-cluster right))
=>        (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
          (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6) (create$ ?*sequence-right-cluster*))
            then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-finish-7
          (declare (salience 57))
          ?f1 <- (opn (number ?n1) (machining_stage finish) (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6 ?n7))
          (operation (number ?n1) (on-feature ?N1))
          (feature (number ?N1) (name EXTERNAL_STEP))
          (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) (not (= ?n7 0)) ))
          (opn (number ?n2)  (setup-cluster right))
          (opn (number ?n3)  (setup-cluster right))
          (opn (number ?n4)  (setup-cluster right))
          (opn (number ?n5)  (setup-cluster right))
          (opn (number ?n6)  (setup-cluster right))
          (opn (number ?n7)  (setup-cluster right))
=>        (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
          (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6 ?n7) (create$ ?*sequence-right-cluster*))
            then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
```

```
;;;;;FACE;;;;;
(defrule MAIN::sequence-right-cluster-finish-2e
            (declare (salience 50))
            ?f1 <- (opn (number ?n1) (machining_stage finish)  (setup-cluster right) (preceding_opn ?n2))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name FACE))
            (test (not (= ?n2 0)))
            (opn (number ?n2)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2) (create$ ?*sequence-right-cluster*))
               then
               (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-finish-3e
            (declare (salience 49))
            ?f1 <- (opn (number ?n1) (machining_stage finish)  (setup-cluster right) (preceding_opn ?n2 ?n3))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name FACE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0))))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3) (create$ ?*sequence-right-cluster*))
               then (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-finish-4e
            (declare (salience 48))
            ?f1 <- (opn (number ?n1) (machining_stage finish)  (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name FACE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0))))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
            (opn (number ?n4)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4) (create$ ?*sequence-right-cluster*))
               then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-finish-5e
            (declare (salience 47))
            ?f1 <- (opn (number ?n1) (machining_stage finish)  (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4 ?n5))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name FACE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) ))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
            (opn (number ?n4)  (setup-cluster right))
            (opn (number ?n5)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5) (create$ ?*sequence-right-cluster*))
               then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))

;;;;;;HOLE;;;;;;
(defrule MAIN::sequence-right-cluster-finish-2a
            (declare (salience 22))
            ?f1 <- (opn (number ?n1) (machining_stage finish) (setup-cluster right) (preceding_opn ?n2))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name HOLE))
            (test (not (= ?n2 0)))
            (opn (number ?n2) (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2) (create$ ?*sequence-right-cluster*))
               then
               (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-finish-3a
            (declare (salience 21))
            ?f1 <- (opn (number ?n1) (machining_stage finish) (setup-cluster right) (preceding_opn ?n2 ?n3))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name HOLE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0))))
            (opn (number ?n2) (setup-cluster right))
            (opn (number ?n3) (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3) (create$ ?*sequence-right-cluster*))
              then (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-finish-4a
            (declare (salience 20))
            ?f1 <- (opn (number ?n1) (machining_stage finish) (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name HOLE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0))))
            (opn (number ?n2) (setup-cluster right))
            (opn (number ?n3) (setup-cluster right))
            (opn (number ?n4) (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4) (create$ ?*sequence-right-cluster*))
              then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-finish-5a
            (declare (salience 19))
            ?f1 <- (opn (number ?n1) (machining_stage finish) (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4 ?n5))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name HOLE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) ))
            (opn (number ?n2) (setup-cluster right))
            (opn (number ?n3) (setup-cluster right))
            (opn (number ?n4) (setup-cluster right))
            (opn (number ?n5) (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5) (create$ ?*sequence-right-cluster*))
              then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-finish-6a
            (declare (salience 18))
            ?f1 <- (opn (number ?n1) (machining_stage finish) (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name HOLE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) ))
            (opn (number ?n2) (setup-cluster right))
            (opn (number ?n3) (setup-cluster right))
            (opn (number ?n4) (setup-cluster right))
            (opn (number ?n5) (setup-cluster right))
            (opn (number ?n6) (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6) (create$ ?*sequence-right-cluster*))
              then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-finish-7a
            (declare (salience 17))
            ?f1 <- (opn (number ?n1) (machining_stage finish) (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6 ?n7))
            (operation (number ?n1) (on-feature ?N1))
            (feature (number ?N1) (name HOLE))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) (not (= ?n7 0)) ))
            (opn (number ?n2) (setup-cluster right))
            (opn (number ?n3) (setup-cluster right))
            (opn (number ?n4) (setup-cluster right))
            (opn (number ?n5) (setup-cluster right))
            (opn (number ?n6) (setup-cluster right))
            (opn (number ?n7) (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6 ?n7) (create$ ?*sequence-right-cluster*))
              then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
```

```
;;;;;;EXTERNAL_TAPER;;;;;;
(defrule MAIN::sequence-right-cluster-finish-2d
        (declare (salience 56))
        ?f1 <- (opn (number ?n1) (machining_stage finish) (setup-cluster right) (preceding_opn ?n2))
        (operation (number ?n1) (on-feature ?N1))
        (feature (number ?N1) (name EXTERNAL_TAPER))
        (test (not (= ?n2 0)))
        (opn (number ?n2) (setup-cluster right))
=>      (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
        (if (subsetp (create$ ?n2) (create$ ?*sequence-right-cluster*))
          then
          (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-finish-3d
        (declare (salience 55))
        ?f1 <- (opn (number ?n1) (machining_stage finish) (setup-cluster right) (preceding_opn ?n2 ?n3))
        (operation (number ?n1) (on-feature ?N1))
        (feature (number ?N1) (name EXTERNAL_TAPER))
        (test (and (not (= ?n2 ?n3)) (not (= ?n2 0)) (not (= ?n3 0))))
        (opn (number ?n2) (setup-cluster right))
        (opn (number ?n3) (setup-cluster right))
=>      (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
        (if (subsetp (create$ ?n2 ?n3) (create$ ?*sequence-right-cluster*))
          then (bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-finish-4d
        (declare (salience 54))
        ?f1 <- (opn (number ?n1) (machining_stage finish) (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4))
        (operation (number ?n1) (on-feature ?N1))
        (feature (number ?N1) (name EXTERNAL_TAPER))
        (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0))))
        (opn (number ?n2) (setup-cluster right))
        (opn (number ?n3) (setup-cluster right))
        (opn (number ?n4) (setup-cluster right))
=>      (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
        (if (subsetp (create$ ?n2 ?n3 ?n4) (create$ ?*sequence-right-cluster*))
          then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-finish-5d
        (declare (salience 53))
        ?f1 <- (opn (number ?n1) (machining_stage finish) (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4 ?n5))
        (operation (number ?n1) (on-feature ?N1))
        (feature (number ?N1) (name EXTERNAL_TAPER))
        (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) ))
        (opn (number ?n2) (setup-cluster right))
        (opn (number ?n3) (setup-cluster right))
        (opn (number ?n4) (setup-cluster right))
        (opn (number ?n5) (setup-cluster right))
=>      (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
        (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5) (create$ ?*sequence-right-cluster*))
          then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-finish-6d
        (declare (salience 52))
        ?f1 <- (opn (number ?n1) (machining_stage finish) (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6))
        (operation (number ?n1) (on-feature ?N1))
        (feature (number ?N1) (name EXTERNAL_TAPER))
        (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) ))
        (opn (number ?n2) (setup-cluster right))
        (opn (number ?n3) (setup-cluster right))
        (opn (number ?n4) (setup-cluster right))
        (opn (number ?n5) (setup-cluster right))
        (opn (number ?n6) (setup-cluster right))
=>      (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
        (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6) (create$ ?*sequence-right-cluster*))
          then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule MAIN::sequence-right-cluster-finish-7d
        (declare (salience 51))
```

```
            ?f1 <- (opn (number ?n1) (machining_stage finish)  (setup-cluster right) (preceding_opn ?n2 ?n3 ?n4 ?n5 ?n6 ?n7))
            (operation (number ?n1)  (on-feature ?N1))
            (feature (number ?N1) (name EXTERNAL_TAPER))
            (test (and (not (= ?n2 ?n3)) (not (= ?n2 ?n4)) (not (= ?n3 ?n4)) (not (= ?n2 0)) (not (= ?n3 0)) (not (= ?n4 0)) (not (= ?n5
0)) (not (= ?n6 0)) (not (= ?n7 0)) ))
            (opn (number ?n2)  (setup-cluster right))
            (opn (number ?n3)  (setup-cluster right))
            (opn (number ?n4)  (setup-cluster right))
            (opn (number ?n5)  (setup-cluster right))
            (opn (number ?n6)  (setup-cluster right))
            (opn (number ?n7)  (setup-cluster right))
=>          (bind ?*opn-right-cluster* (fact-slot-value ?f1 number))
            (if (subsetp (create$ ?n2 ?n3 ?n4 ?n5 ?n6 ?n7) (create$ ?*sequence-right-cluster*))
               then(bind ?*sequence-right-cluster* (create$ ?*sequence-right-cluster* ?*opn-right-cluster*))))
(defrule report-sequence-setup-clusters-right
=>          (open "f:\\Sankha\\personal\\Research\\Actual\\clips_appl\\setup_planning_es\\pc_generation\\sequence_r.clp"
sequence_r "w")
            (printout sequence_r ?*sequence-right-cluster*)
            (close sequence_r)
            (printout t "The sequence of operations from the right is " ?*sequence-right-cluster* crlf))
(defrule save_to_file
=>          (save-facts facts3.clp visible))

;;; Note that the results of execution of all the above rules are stored in the datafile, called facts3.clp
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; file name: program_file_4.CLP;;;;;;;;;;;;;;;;;;;;;;;
;;;==================================================================================
;;;   Expert System for selection of locating and clamping surfaces for each set-up
;;;   CLIPS Version 6.10
;;;   To execute, merely load this program, then load the file,"facts1.clp", that is generated by "rules_Set_1.clp", then reset and run.
;;;==================================================================================
(defmodule MAIN
 (export deftemplate ?ALL))

;;;*****************************
;;;* DEFTEMPLATE DEFINITIONS *
;;;*****************************

(deftemplate MAIN::feature
        (slot number (type INTEGER) (default ?NONE))
        (slot name (type SYMBOL) (allowed-symbols CHAMFER EXTERNAL_STEP FACE GROOVE HOLE KEYWAY
EXTERNAL_TAPER THREAD HOLE))
        (slot type (type SYMBOL) (allowed-symbols EXTERNAL INTERNAL))
        (slot subtype (type SYMBOL) (allowed-symbols PRIMARY SECONDARY))
        (slot secondary_feature_to (type INTEGER) (default ?DERIVE))
        (multislot adjacent_features (type INTEGER) (default ?DERIVE))
        (multislot adjacent_features_names (type SYMBOL) (allowed-symbols CHAMFER EXTERNAL_STEP FACE
GROOVE HOLE KEYWAY EXTERNAL_TAPER THREAD HOLE))
        (multislot reference_features (type INTEGER) (default 0))
        (slot step_diameter (type NUMBER))
        (multislot adjacent_step_diameters (type NUMBER))
        (slot hole_diameter (type NUMBER))
        (slot hole_depth (type NUMBER))
        (multislot adjacent_hole_diameters (type NUMBER))
        (multislot adjacent_hole_depths (type NUMBER))
        (slot TAD (type SYMBOL) (allowed-symbols left right right-left) (default ?NONE)))
(deftemplate MAIN::operation
        (slot number (type INTEGER) (default ?NONE))
        (slot type (type SYMBOL) (default ?NONE))
        (slot machining_stage (type SYMBOL) (allowed-symbols rough semifinish finish))
        (slot on-feature (type INTEGER) (default ?NONE))
        (slot TAD (type SYMBOL) (allowed-symbols left right right-left) (default ?NONE))
        (multislot relation-with-feature (type INTEGER) (default 0))
        (multislot tolerance (type NUMBER) (default 0)))
(deftemplate MAIN::setup-from-left-cluster
        (multislot operation_numbers))
(deftemplate MAIN::setup-from-right-cluster
        (multislot operation_numbers))
(deftemplate MAIN::operation_precedence
        (multislot values))
(deftemplate MAIN::opn
        (slot number (type INTEGER) (default ?NONE))
        (slot type (type SYMBOL))
        (slot machining_stage (type SYMBOL) (allowed-symbols rough semifinish finish))
        (slot setup-cluster (type SYMBOL) (allowed-symbols left right))
        (multislot preceding_opn (type INTEGER) (default 0)))
(deftemplate MAIN::order-of-features-from-left-external-to-internal
        (multislot feature-numbers (type INTEGER) (default ?NONE)))
(deftemplate MAIN::order-of-features-from-right-external-to-internal
        (multislot feature-numbers (type INTEGER) (default ?NONE)))
(deftemplate MAIN::feature-with-largest-dia
        (slot number (type INTEGER) (default ?NONE)))
(deftemplate MAIN::datums_selected
        (slot clamping_surface (type INTEGER))
        (slot locating_surface (type INTEGER))
        (slot setup (type SYMBOL) (allowed-symbols left right)))

;;;*****************************
;;;* DEFFUNCTION DEFINITIONS *
;;;*****************************
```

```
(deffunction MAIN::feature-with-tightest-tolerance (?f1)
            (bind $?num (fact-slot-value ?f1 tolerance))
            (while (>= (length$ $?num) 2)
              (bind ?num1 (first$ $?num))
              (bind $?num (rest$ $?num))
              (bind ?num2 (first$ $?num))
              (bind ?num11 (nth$ 1 (explode$ (implode$ ?num1))))
              (bind ?num22 (nth$ 1 (explode$ (implode$ ?num2))))
              (if (> ?num11 ?num22)
                then (bind ?num1 ?num2)
              else (bind ?num1 ?num1)))
            (bind ?num3 (member$ (explode$ (implode$ ?num1)) (fact-slot-value ?f1 tolerance)))
            (bind ?num4 (nth$ ?num3 (fact-slot-value ?f1 relation-with-feature)))
            return ?num4)
(deffunction MAIN::feature-with-tightest-tolerance-case-of-one-feature (?f1)
            (bind ?num5 (nth$ 1 (explode$ (implode$ (fact-slot-value ?f1 relation-with-feature)))))
            return ?num5)

;;;***************************************************************
;;;* RULES FOR DETERMINING LOCATING AND CLAMPING SURFACES*
;;;***************************************************************

(defrule MAIN::locating-and-clamping-surfaces-left-one-feature0a
            (declare (salience 90))
            (feature (number ?c) (name EXTERNAL_STEP) (adjacent_features $? ?d $?))
            ?f1 <- (operation (on-feature ?c) (TAD left) (machining_stage rough))
            (feature (number ?d) (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER) (adjacent_features $? ?c $?))
            (operation (on-feature ?d) (TAD left) (machining_stage rough))
            (test (= (length$ (fact-slot-value ?f1 tolerance)) 1))
            (operation (on-feature =(feature-with-tightest-tolerance-case-of-one-feature ?f1)) (TAD right) (machining_stage rough))
            (feature (number =(feature-with-tightest-tolerance-case-of-one-feature ?f1)) (name EXTERNAL_STEP)
(adjacent_features $? ?e $?))
            (feature (number ?e) (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER) (adjacent_features $? =(feature-with-
tightest-tolerance-case-of-one-feature ?f1) $?))
=>          (assert (datums_selected (setup left) (clamping_surface =(feature-with-tightest-tolerance-case-of-one-feature ?f1))
(locating_surface =(feature-with-tightest-tolerance-case-of-one-feature ?f1)))))
(defrule MAIN::locating-and-clamping-surfaces-left-one-feature0aa
            (declare (salience 90))
            (feature (number ?c) (name EXTERNAL_STEP) (adjacent_features $? ?d $?))
            ?f1 <- (operation (on-feature ?c) (TAD left) (machining_stage rough))
            (feature (number ?d) (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER) (adjacent_features $? ?c $?))
            (operation (on-feature ?d) (TAD left) (machining_stage rough))
            (test (= (length$ (fact-slot-value ?f1 tolerance)) 1))
            (operation (on-feature =(feature-with-tightest-tolerance-case-of-one-feature ?f1)) (TAD right) (machining_stage rough))
            (feature (number =(feature-with-tightest-tolerance-case-of-one-feature ?f1)) (name EXTERNAL_STEP)
(adjacent_features $? ?e $?))
            (feature (number ?e) (name FACE) (adjacent_features $? =(feature-with-tightest-tolerance-case-of-one-feature ?f1) $?))

=>          (assert (datums_selected (setup left) (clamping_surface =(feature-with-tightest-tolerance-case-of-one-feature ?f1))
(locating_surface ?e))))
(defrule MAIN::locating-and-clamping-surfaces-left-one-feature0b
            (declare (salience 90))
            (feature (number ?c) (name EXTERNAL_STEP) (adjacent_features $? ?d $?))
            ?f1 <- (operation (on-feature ?c) (TAD left) (machining_stage rough))
            (feature (number ?d) (name FACE) (adjacent_features $? ?c $?))
            (operation (on-feature ?d) (TAD left) (machining_stage rough))
            (test (= (length$ (fact-slot-value ?f1 tolerance)) 1))
            (operation (on-feature =(feature-with-tightest-tolerance-case-of-one-feature ?f1)) (TAD right) (machining_stage rough))
            (feature (number =(feature-with-tightest-tolerance-case-of-one-feature ?f1)) (name EXTERNAL_STEP)
(adjacent_features $? ?e $?))
            (feature (number ?e) (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER) (adjacent_features $? =(feature-with-
tightest-tolerance-case-of-one-feature ?f1) $?))
=>          (assert (datums_selected (setup left) (clamping_surface =(feature-with-tightest-tolerance-case-of-one-feature ?f1))
(locating_surface =(feature-with-tightest-tolerance-case-of-one-feature ?f1)))))
(defrule MAIN::locating-and-clamping-surfaces-left-one-feature0bb
            (declare (salience 90))
```

```
        (feature (number ?c) (name EXTERNAL_STEP) (adjacent_features $? ?d $?))
        ?f1 <- (operation (on-feature ?c) (TAD left) (machining_stage rough))
        (feature (number ?d) (name FACE) (adjacent_features $? ?c $?))
        (operation (on-feature ?d) (TAD left) (machining_stage rough))
        (test (= (length$ (fact-slot-value ?f1 tolerance)) 1))
        (operation (on-feature =(feature-with-tightest-tolerance-case-of-one-feature ?f1)) (TAD right) (machining_stage rough))
        (feature (number =(feature-with-tightest-tolerance-case-of-one-feature ?f1)) (name EXTERNAL_STEP)
(adjacent_features $? ?e $?))
        (feature (number ?e) (name FACE) (adjacent_features $? =(feature-with-tightest-tolerance-case-of-one-feature ?f1) $?))

=>      (assert (datums_selected (setup left) (clamping_surface =(feature-with-tightest-tolerance-case-of-one-feature ?f1))
(locating_surface ?e))))
(defrule MAIN::locating-and-clamping-surfaces-right-one-feature0a
        (declare (salience 90))
        (feature (number ?c) (name EXTERNAL_STEP) (adjacent_features $? ?d $?))
        ?f1 <- (operation (on-feature ?c) (TAD right) (machining_stage rough))
        (feature (number ?d) (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER) (adjacent_features $? ?c $?))
        (operation (on-feature ?d) (TAD right) (machining_stage rough))
        (test (= (length$ (fact-slot-value ?f1 tolerance)) 1))
        (operation (on-feature =(feature-with-tightest-tolerance-case-of-one-feature ?f1)) (TAD left) (machining_stage rough))
        (feature (number =(feature-with-tightest-tolerance-case-of-one-feature ?f1)) (name EXTERNAL_STEP)
(adjacent_features $? ?e $?))
        (feature (number ?e) (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER) (adjacent_features $? =(feature-with-
tightest-tolerance-case-of-one-feature ?f1) $?))
=>      (assert (datums_selected (setup right) (clamping_surface =(feature-with-tightest-tolerance-case-of-one-feature ?f1))
(locating_surface =(feature-with-tightest-tolerance-case-of-one-feature ?f1)))))
(defrule MAIN::locating-and-clamping-surfaces-right-one-feature0aa
        (declare (salience 90))
        (feature (number ?c) (name EXTERNAL_STEP) (adjacent_features $? ?d $?))
        ?f1 <- (operation (on-feature ?c) (TAD right) (machining_stage rough))
        (feature (number ?d) (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER) (adjacent_features $? ?c $?))
        (operation (on-feature ?d) (TAD right) (machining_stage rough))
        (test (= (length$ (fact-slot-value ?f1 tolerance)) 1))
        (operation (on-feature =(feature-with-tightest-tolerance-case-of-one-feature ?f1)) (TAD left) (machining_stage rough))
        (feature (number =(feature-with-tightest-tolerance-case-of-one-feature ?f1)) (name EXTERNAL_STEP)
(adjacent_features $? ?e $?))
        (feature (number ?e) (name FACE) (adjacent_features $? =(feature-with-tightest-tolerance-case-of-one-feature ?f1) $?))

=>      (assert (datums_selected (setup right) (clamping_surface =(feature-with-tightest-tolerance-case-of-one-feature ?f1))
(locating_surface ?e))))

(defrule MAIN::locating-and-clamping-surfaces-right-one-feature0b
        (declare (salience 90))
        (feature (number ?c) (name EXTERNAL_STEP) (adjacent_features $? ?d $?))
        ?f1 <- (operation (on-feature ?c) (TAD right) (machining_stage rough))
        (feature (number ?d) (name FACE) (adjacent_features $? ?c $?))
        (operation (on-feature ?d) (TAD right) (machining_stage rough))
        (test (= (length$ (fact-slot-value ?f1 tolerance)) 1))
        (operation (on-feature =(feature-with-tightest-tolerance-case-of-one-feature ?f1)) (TAD left) (machining_stage rough))
        (feature (number =(feature-with-tightest-tolerance-case-of-one-feature ?f1)) (name EXTERNAL_STEP)
(adjacent_features $? ?e $?))
        (feature (number ?e) (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER) (adjacent_features $? =(feature-with-
tightest-tolerance-case-of-one-feature ?f1) $?))
=>      (assert (datums_selected (setup right) (clamping_surface =(feature-with-tightest-tolerance-case-of-one-feature ?f1))
(locating_surface =(feature-with-tightest-tolerance-case-of-one-feature ?f1)))))
(defrule MAIN::locating-and-clamping-surfaces-right-one-feature0bb
        (declare (salience 90))
        (feature (number ?c) (name EXTERNAL_STEP) (adjacent_features $? ?d $?))
        ?f1 <- (operation (on-feature ?c) (TAD right) (machining_stage rough))
        (feature (number ?d) (name FACE) (adjacent_features $? ?c $?))
        (operation (on-feature ?d) (TAD right) (machining_stage rough))
        (test (= (length$ (fact-slot-value ?f1 tolerance)) 1))
        (operation (on-feature =(feature-with-tightest-tolerance-case-of-one-feature ?f1)) (TAD left) (machining_stage rough))
        (feature (number =(feature-with-tightest-tolerance-case-of-one-feature ?f1)) (name EXTERNAL_STEP)
(adjacent_features $? ?e $?))
```

```
		(feature (number ?e) (name FACE) (adjacent_features $? =(feature-with-tightest-tolerance-case-of-one-feature ?f1) $?))

=>		(assert (datums_selected (setup right) (clamping_surface =(feature-with-tightest-tolerance-case-of-one-feature ?f1))
(locating_surface ?e))))
(defrule MAIN::locating-and-clamping-surfaces-left-multiple-features0a
		(declare (salience 90))
		(feature (number ?c) (name EXTERNAL_STEP) (adjacent_features $? ?d $?))
		?f1 <- (operation (on-feature ?c) (TAD left) (machining_stage rough))
		(feature (number ?d) (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER) (adjacent_features $? ?c $?))
		(operation (on-feature ?d) (TAD left) (machining_stage rough))
		(test (>= (length$ (fact-slot-value ?f1 tolerance)) 2))
		(operation (on-feature =(feature-with-tightest-tolerance ?f1)) (TAD right) (machining_stage rough))
		(feature (number =(feature-with-tightest-tolerance ?f1)) (name EXTERNAL_STEP) (adjacent_features $? ?e $?))
		(feature (number ?e) (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER) (adjacent_features $? =(feature-with-
tightest-tolerance ?f1) $?))
=>		(assert (datums_selected (setup left) (clamping_surface =(feature-with-tightest-tolerance ?f1)) (locating_surface
=(feature-with-tightest-tolerance ?f1)))))
(defrule MAIN::locating-and-clamping-surfaces-left-multiple-features0aa
		(declare (salience 90))
		(feature (number ?c) (name EXTERNAL_STEP) (adjacent_features $? ?d $?))
		?f1 <- (operation (on-feature ?c) (TAD left) (machining_stage rough))
		(feature (number ?d) (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER) (adjacent_features $? ?c $?))
		(operation (on-feature ?d) (TAD left) (machining_stage rough))
		(test (>= (length$ (fact-slot-value ?f1 tolerance)) 2))
		(operation (on-feature =(feature-with-tightest-tolerance ?f1)) (TAD right) (machining_stage rough))
		(feature (number =(feature-with-tightest-tolerance ?f1)) (name EXTERNAL_STEP) (adjacent_features $? ?e $?))
		(feature (number ?e) (name FACE) (adjacent_features $? =(feature-with-tightest-tolerance ?f1) $?))
=>		(assert (datums_selected (setup left) (clamping_surface =(feature-with-tightest-tolerance ?f1)) (locating_surface ?e))))
(defrule MAIN::locating-and-clamping-surfaces-left-multiple-features0b
		(declare (salience 90))
		(feature (number ?c) (name EXTERNAL_STEP) (adjacent_features $? ?d $?))
		?f1 <- (operation (on-feature ?c) (TAD left) (machining_stage rough))
		(feature (number ?d) (name FACE) (adjacent_features $? ?c $?))
		(operation (on-feature ?d) (TAD left) (machining_stage rough))
		(test (>= (length$ (fact-slot-value ?f1 tolerance)) 2))
		(operation (on-feature =(feature-with-tightest-tolerance ?f1)) (TAD right) (machining_stage rough))
		(feature (number =(feature-with-tightest-tolerance ?f1)) (name EXTERNAL_STEP) (adjacent_features $? ?e $?))
		(feature (number ?e) (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER) (adjacent_features $? =(feature-with-
tightest-tolerance ?f1) $?))
=>		(assert (datums_selected (setup left) (clamping_surface =(feature-with-tightest-tolerance ?f1)) (locating_surface
=(feature-with-tightest-tolerance ?f1)))))
(defrule MAIN::locating-and-clamping-surfaces-left-multiple-features0bb
		(declare (salience 90))
		(feature (number ?c) (name EXTERNAL_STEP) (adjacent_features $? ?d $?))
		?f1 <- (operation (on-feature ?c) (TAD left) (machining_stage rough))
		(feature (number ?d) (name FACE) (adjacent_features $? ?c $?))
		(operation (on-feature ?d) (TAD left) (machining_stage rough))
		(test (>= (length$ (fact-slot-value ?f1 tolerance)) 2))
		(operation (on-feature =(feature-with-tightest-tolerance ?f1)) (TAD right) (machining_stage rough))
		(feature (number =(feature-with-tightest-tolerance ?f1)) (name EXTERNAL_STEP) (adjacent_features $? ?e $?))
		(feature (number ?e) (name FACE) (adjacent_features $? =(feature-with-tightest-tolerance ?f1) $?))
=>		(assert (datums_selected (setup left) (clamping_surface =(feature-with-tightest-tolerance ?f1)) (locating_surface ?e))))
(defrule MAIN::locating-and-clamping-surfaces-right-multiple-features0a
		(declare (salience 90))
		(feature (number ?c) (name EXTERNAL_STEP) (adjacent_features $? ?d $?))
		?f1 <- (operation (on-feature ?c) (TAD right) (machining_stage rough))
		(feature (number ?d) (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER) (adjacent_features $? ?c $?))
		(operation (on-feature ?d) (TAD right) (machining_stage rough))
		(test (>= (length$ (fact-slot-value ?f1 tolerance)) 2))
		(operation (on-feature =(feature-with-tightest-tolerance ?f1)) (TAD left) (machining_stage rough))
		(feature (number =(feature-with-tightest-tolerance ?f1)) (name EXTERNAL_STEP) (adjacent_features $? ?e $?))
		(feature (number ?e) (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER) (adjacent_features $? =(feature-with-
tightest-tolerance ?f1) $?))
=>		(assert (datums_selected (setup right) (clamping_surface =(feature-with-tightest-tolerance ?f1)) (locating_surface
=(feature-with-tightest-tolerance ?f1)))))
(defrule MAIN::locating-and-clamping-surfaces-right-multiple-features0aa
            (declare (salience 90))
            (feature (number ?c) (name EXTERNAL_STEP) (adjacent_features $? ?d $?))
            ?f1 <- (operation (on-feature ?c) (TAD right) (machining_stage rough))
            (feature (number ?d) (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER) (adjacent_features $? ?c $?))
            (operation (on-feature ?d) (TAD right) (machining_stage rough))
            (test (>= (length$ (fact-slot-value ?f1 tolerance)) 2))
            (operation (on-feature =(feature-with-tightest-tolerance ?f1)) (TAD left) (machining_stage rough))
            (feature (number =(feature-with-tightest-tolerance ?f1)) (name EXTERNAL_STEP) (adjacent_features $? ?e $?))
            (feature (number ?e) (name FACE) (adjacent_features $? =(feature-with-tightest-tolerance ?f1) $?))
=>          (assert (datums_selected (setup right) (clamping_surface =(feature-with-tightest-tolerance ?f1)) (locating_surface ?e))))
(defrule MAIN::locating-and-clamping-surfaces-right-multiple-features0b
            (declare (salience 90))
            (feature (number ?c) (name EXTERNAL_STEP) (adjacent_features $? ?d $?))
            ?f1 <- (operation (on-feature ?c) (TAD right) (machining_stage rough))
            (feature (number ?d) (name FACE) (adjacent_features $? ?c $?))
            (operation (on-feature ?d) (TAD right) (machining_stage rough))
            (test (>= (length$ (fact-slot-value ?f1 tolerance)) 2))
            (operation (on-feature =(feature-with-tightest-tolerance ?f1)) (TAD left) (machining_stage rough))
            (feature (number =(feature-with-tightest-tolerance ?f1)) (name EXTERNAL_STEP) (adjacent_features $? ?e $?))
            (feature (number ?e) (name ?name&EXTERNAL_STEP|EXTERNAL_TAPER) (adjacent_features $? =(feature-with-
tightest-tolerance ?f1) $?))
=>          (assert (datums_selected (setup right) (clamping_surface =(feature-with-tightest-tolerance ?f1)) (locating_surface
=(feature-with-tightest-tolerance ?f1)))))
(defrule MAIN::locating-and-clamping-surfaces-right-multiple-features0bb
            (declare (salience 90))
            (feature (number ?c) (name EXTERNAL_STEP) (adjacent_features $? ?d $?))
            ?f1 <- (operation (on-feature ?c) (TAD right) (machining_stage rough))
            (feature (number ?d) (name FACE) (adjacent_features $? ?c $?))
            (operation (on-feature ?d) (TAD right) (machining_stage rough))
            (test (>= (length$ (fact-slot-value ?f1 tolerance)) 2))
            (operation (on-feature =(feature-with-tightest-tolerance ?f1)) (TAD left) (machining_stage rough))
            (feature (number =(feature-with-tightest-tolerance ?f1)) (name EXTERNAL_STEP) (adjacent_features $? ?e $?))
            (feature (number ?e) (name FACE) (adjacent_features $? =(feature-with-tightest-tolerance ?f1) $?))
=>          (assert (datums_selected (setup right) (clamping_surface =(feature-with-tightest-tolerance ?f1)) (locating_surface ?e))))
(defrule MAIN::locating-and-clamping-surfaces-left-no-relation1
            (declare (salience 0))
            (not (operation (TAD right) (relation-with-feature ~0)))
            (feature (number ?a) (type EXTERNAL) (name EXTERNAL_STEP) (adjacent_features $? ?b $?))
            (feature-with-largest-dia (number ?a))
            (operation (on-feature ?a) (machining_stage rough) (TAD left))
            (feature (number ?b) (type EXTERNAL) (name EXTERNAL_STEP) (TAD left) (adjacent_features $? ?a $?))
=>          (assert (datums_selected (setup right) (clamping_surface ?a) (locating_surface ?a))))
(defrule MAIN::locating-and-clamping-surfaces-left-no-relation2
            (declare (salience 0))
            (not (operation (TAD right) (relation-with-feature ~0)))
            (feature (number ?a) (type EXTERNAL) (name EXTERNAL_STEP) (adjacent_features $? ?b $?))
            (feature-with-largest-dia (number ?a))
            (operation (on-feature ?a) (machining_stage rough) (TAD left))
            (feature (number ?b) (type EXTERNAL) (name FACE) (TAD left) (adjacent_features $? ?a $?))
=>          (assert (datums_selected (setup right) (clamping_surface ?a) (locating_surface ?b))))
(defrule MAIN::locating-and-clamping-surfaces-left-no-relation3
            (declare (salience 0))
            (not (operation (TAD right) (relation-with-feature ~0)))
            (feature (number ?a) (type EXTERNAL) (name EXTERNAL_STEP) (adjacent_features $? ?b $?))
            (feature-with-largest-dia (number ?a))
            (feature (number ?b) (type EXTERNAL) (name ?name&~FACE&~EXTERNAL_STEP) (TAD left) (adjacent_features
$? ?a $?))
            (feature (number ?c) (name EXTERNAL_STEP) (TAD left) (adjacent_features $? ?d $?) (step_diameter ?d1)
(adjacent_step_diameters $? ?d2 $?))
            (feature (number ?d) (name EXTERNAL_STEP) (TAD left) (adjacent_features $? ?c $?) (step_diameter ?d2)
(adjacent_step_diameters $? ?d1 $?))
            (order-of-features-from-left-external-to-internal (feature-numbers $? ?a ?b $? ?c ?d $?))
            (test (not (= ?a ?c)))
            (test (> ?d1 ?d2))
=>          (assert (datums_selected (setup right) (clamping_surface ?c) (locating_surface ?c))))
(defrule MAIN::locating-and-clamping-surfaces-left-no-relation4
            (declare (salience 0))
            (not (operation (TAD right) (relation-with-feature ~0)))
            (feature (number ?a) (type EXTERNAL) (name EXTERNAL_STEP) (adjacent_features $? ?b $?))
            (feature-with-largest-dia (number ?a))
            (feature (number ?b) (type EXTERNAL) (name ?name&~FACE&~EXTERNAL_STEP) (TAD left) (adjacent_features
$? ?a $?))
            (feature (number ?c) (name EXTERNAL_STEP) (TAD left) (adjacent_features $? ?d $?) (step_diameter ?d1)
(adjacent_step_diameters $? ?d2 $?))
            (feature (number ?d) (name FACE) (TAD left) (adjacent_features $? ?c $?))
            (order-of-features-from-left-external-to-internal (feature-numbers $? ?a ?b $? ?c ?d $?))
            (test (not (= ?a ?c)))
=>          (assert (datums_selected (setup right) (clamping_surface ?c) (locating_surface ?d))))
(defrule MAIN::locating-and-clamping-surfaces-right-no-relation1
            (declare (salience 0))
            (not (operation (TAD left) (relation-with-feature ~0)))
            (feature (number ?a) (type EXTERNAL) (name EXTERNAL_STEP) (adjacent_features $? ?b $?))
            (feature-with-largest-dia (number ?a))
            (operation (on-feature ?a) (machining_stage rough) (TAD right))
            (feature (number ?b) (type EXTERNAL) (name EXTERNAL_STEP) (TAD right) (adjacent_features $? ?a $?))
=>          (assert (datums_selected (setup left) (clamping_surface ?a) (locating_surface ?a))))

(defrule MAIN::locating-and-clamping-surfaces-right-no-relation2
            (declare (salience 0))
            (not (operation (TAD left) (relation-with-feature ~0)))
            (feature (number ?a) (type EXTERNAL) (name EXTERNAL_STEP) (adjacent_features $? ?b $?))
            (feature-with-largest-dia (number ?a))
            (operation (on-feature ?a) (machining_stage rough) (TAD right))
            (feature (number ?b) (type EXTERNAL) (name FACE) (TAD right) (adjacent_features $? ?a $?))
=>          (assert (datums_selected (setup left) (clamping_surface ?a) (locating_surface ?b))))
(defrule MAIN::locating-and-clamping-surfaces-right-no-relation3
            (declare (salience 0))
            (not (operation (TAD left) (relation-with-feature ~0)))
            (feature (number ?a) (type EXTERNAL) (name EXTERNAL_STEP) (adjacent_features $? ?b $?))
            (feature-with-largest-dia (number ?a))
            (feature (number ?b) (type EXTERNAL) (name ?name&~FACE&~EXTERNAL_STEP) (TAD right) (adjacent_features
$? ?a $?))
            (feature (number ?c) (name EXTERNAL_STEP) (TAD right) (adjacent_features $? ?d $?) (step_diameter ?d1)
(adjacent_step_diameters $? ?d2 $?))
            (feature (number ?d) (name EXTERNAL_STEP) (TAD right) (adjacent_features $? ?c $?) (step_diameter ?d2)
(adjacent_step_diameters $? ?d1 $?))
            (order-of-features-from-right-external-to-internal (feature-numbers $? ?d ?c $? ?b ?a $?))
            (test (not (= ?a ?c)))
            (test (> ?d1 ?d2))
=>          (assert (datums_selected (setup left) (clamping_surface ?c) (locating_surface ?c))))
(defrule MAIN::locating-and-clamping-surfaces-right-no-relation4
            (declare (salience 0))
            (not (operation (TAD left) (relation-with-feature ~0)))
            (feature (number ?a) (type EXTERNAL) (name EXTERNAL_STEP) (adjacent_features $? ?b $?))
            (feature-with-largest-dia (number ?a))
            (feature (number ?b) (type EXTERNAL) (name ?name&~FACE&~EXTERNAL_STEP) (TAD right) (adjacent_features
$? ?a $?))
            (feature (number ?c) (name EXTERNAL_STEP) (TAD right) (adjacent_features $? ?d $?) (step_diameter ?d1)
(adjacent_step_diameters $? ?d2 $?))
            (feature (number ?d) (name FACE) (TAD right) (adjacent_features $? ?c $?))
            (order-of-features-from-right-external-to-internal (feature-numbers $? ?d ?c $? ?b ?a $?))
            (test (not (= ?a ?c)))
=>          (assert (datums_selected (setup left) (clamping_surface ?c) (locating_surface ?d))))
(defrule save_to_file
=>          (save-facts facts4.clp visible))

;;; Note that the results of execution of all the above rules are stored in the datafile, called facts4.clp
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; file name: datafile.clp (for illustrative example 1);;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;********************************
;;;* DEFFACTS KNOWLEDGE BASE *
;;;********************************

(deffacts MAIN::features_list
          (feature (number 1) (name FACE) (type EXTERNAL)  (subtype PRIMARY) (adjacent_features 2 15)
(adjacent_features_names EXTERNAL_STEP HOLE) (TAD left))
          (feature (number 2) (name EXTERNAL_STEP) (type EXTERNAL)  (subtype PRIMARY) (adjacent_features 1 3)
(adjacent_features_names FACE FACE)(step_diameter 30)(TAD left))
          (feature (number 3) (name FACE) (type EXTERNAL)  (subtype PRIMARY) (adjacent_features 2 4)
(adjacent_features_names EXTERNAL_STEP EXTERNAL_STEP)(TAD left))
          (feature (number 4) (name EXTERNAL_STEP) (type EXTERNAL)  (subtype PRIMARY) (adjacent_features 3 5)
(adjacent_features_names FACE EXTERNAL_TAPER)(step_diameter 49)(TAD right-left))
          (feature (number 5) (name EXTERNAL_TAPER) (type EXTERNAL)  (subtype PRIMARY) (adjacent_features 4 6)
(adjacent_features_names EXTERNAL_STEP EXTERNAL_STEP)(TAD right))
          (feature (number 6) (name EXTERNAL_STEP) (type EXTERNAL)  (subtype PRIMARY) (adjacent_features 5 7)
(adjacent_features_names EXTERNAL_TAPER EXTERNAL_STEP)(step_diameter 26)(adjacent_step_diameters 22)(TAD right))
          (feature (number 7) (name EXTERNAL_STEP) (type EXTERNAL)  (subtype PRIMARY) (adjacent_features 6 8)
(adjacent_features_names EXTERNAL_STEP EXTERNAL_STEP)(step_diameter 22)(adjacent_step_diameters 26 15)(TAD right))
          (feature (number 8) (name EXTERNAL_STEP) (type EXTERNAL)  (subtype PRIMARY) (adjacent_features 7 9)
(adjacent_features_names EXTERNAL_STEP EXTERNAL_STEP)(step_diameter 15)(adjacent_step_diameters 22 21)(TAD right))
          (feature (number 9) (name EXTERNAL_STEP) (type EXTERNAL)  (subtype PRIMARY) (adjacent_features 8 10)
(adjacent_features_names EXTERNAL_STEP GROOVE)(step_diameter 21)(adjacent_step_diameters 15)(TAD right))
          (feature (number 10) (name GROOVE) (type EXTERNAL)  (subtype PRIMARY) (adjacent_features 9 11)
(adjacent_features_names EXTERNAL_STEP THREAD)(TAD right))
          (feature (number 111) (name EXTERNAL_STEP) (type EXTERNAL)  (subtype PRIMARY) (adjacent_features 10 12)
(adjacent_features_names GROOVE GROOVE)(step_diameter 19.6)(TAD right))
          (feature (number 11) (name THREAD) (type EXTERNAL)  (subtype SECONDARY) (secondary_feature_to 111)
(adjacent_features 10 12) (adjacent_features_names GROOVE GROOVE)(TAD right))
          (feature (number 12) (name GROOVE) (type EXTERNAL)  (subtype PRIMARY) (adjacent_features 11 13)
(adjacent_features_names THREAD EXTERNAL_STEP)(TAD right))
          (feature (number 13) (name EXTERNAL_STEP) (type EXTERNAL)  (subtype PRIMARY) (adjacent_features 12 14)
(adjacent_features_names GROOVE FACE)(step_diameter 17)(TAD right))
          (feature (number 14) (name FACE) (type EXTERNAL)  (subtype PRIMARY) (adjacent_features 13 18)
(adjacent_features_names EXTERNAL_STEP HOLE) (reference_features 0)(TAD right))
          (feature (number 15) (name HOLE) (type INTERNAL) (subtype PRIMARY) (hole_diameter 18) (hole_depth 10)
(adjacent_features 1 16) (adjacent_features_names FACE HOLE) (adjacent_hole_diameters 15) (adjacent_hole_depths 33)(TAD
left))
          (feature (number 16) (name HOLE) (type INTERNAL)  (subtype PRIMARY) (hole_diameter 15) (hole_depth 33)
(adjacent_features 15 17) (adjacent_features_names HOLE HOLE) (adjacent_hole_diameters 18 10) (adjacent_hole_depths 10
68)(TAD left))
          (feature (number 17) (name HOLE) (type INTERNAL)  (subtype PRIMARY)  (hole_diameter 8) (hole_depth 68)
(adjacent_features 16 18) (adjacent_features_names HOLE HOLE) (adjacent_hole_diameters 15 10) (adjacent_hole_depths 33
75)(TAD right-left))
          (feature (number 18) (name HOLE) (type INTERNAL)  (subtype PRIMARY)  (hole_diameter 10) (hole_depth 75)
(adjacent_features 17 14) (adjacent_features_names HOLE FACE) (adjacent_hole_diameters 8) (adjacent_hole_depths 68)(TAD
right))
          (order-of-features-from-left-external-to-internal (feature-numbers 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18))
          (order-of-features-from-right-external-to-internal (feature-numbers 14 13 12 11 10 9 8 7 6 5 4 3 2 1 18 17 16 15))
          (feature-with-largest-dia (number 4)))

(deffacts MAIN::machining-operations
          (operation (number 101) (type turn-face) (machining_stage rough) (on-feature 1)  (TAD left) (relation-with-feature 0))
          (operation (number 102)(type turn-face) (machining_stage semifinish)(on-feature 1) (TAD left)(relation-with-feature 0))
          (operation (number 103)(type turn-face)(machining_stage finish) (on-feature 1)  (TAD left)(relation-with-feature 0))
          (operation (number 201)  (type turn)  (machining_stage rough)   (on-feature 2)  (TAD left)      (relation-with-feature 0))
          (operation (number 202)  (type turn)  (machining_stage semifinish)(on-feature 2)  (TAD left)(relation-with-feature 0))
          (operation (number 203)  (type finish-turn)  (machining_stage finish)(on-feature 2)  (TAD left) (relation-with-feature 0))
          (operation (number 301) (type turn-face)  (machining_stage rough) (on-feature 3)  (TAD left) (relation-with-feature 0))
          (operation (number 302) (type turn-face)  (machining_stage semifinish)(on-feature 3) (TAD left)(relation-with-feature 0))
          (operation (number 303) (type turn-face)(machining_stage finish)  (on-feature 3)  (TAD left)   (relation-with-feature 0))
          (operation (number 401)(type turn)(machining_stage rough)(on-feature 4)(TAD right-left)(relation-with-feature 2 13 7)
(tolerance 0.1 0.2 0.3))
          (operation (number 402) (type turn)  (machining_stage semifinish)(on-feature 4)(TAD right-left)(relation-with-feature 0))
          (operation (number 403) (type turn)(machining_stage finish)  (on-feature 4)  (TAD right-left) (relation-with-feature 0))
(operation (number 501) (type turn-taper) (machining_stage rough) (on-feature 5)  (TAD right) (relation-with-feature 0))
(operation (number 502) (type turn-taper)(machining_stage semifinish)(on-feature 5)(TAD right)(relation-with-feature 0))
(operation (number 503)  (type turn-taper) (machining_stage finish) (on-feature 5)  (TAD right) (relation-with-feature 0))
(operation (number 601)  (type turn)  (machining_stage rough)  (on-feature 6)  (TAD right)  (relation-with-feature 0))
(operation (number 602)  (type turn)  (machining_stage semifinish)   (on-feature 6)(TAD right)(relation-with-feature 0))
(operation (number 603)  (type turn)  (machining_stage finish)   (on-feature 6)  (TAD right)  (relation-with-feature 0))
(operation (number 701)  (type  turn)  (machining_stage rough)    (on-feature 7)  (TAD right)(relation-with-feature 0))
(operation (number 702)  (type  turn)  (machining_stage semifinish)  (on-feature 7)(TAD right)(relation-with-feature 0))
(operation (number 703)  (type  turn)  (machining_stage finish)    (on-feature 7)  (TAD right)  (relation-with-feature 0))
(operation (number 801)  (type turn)  (machining_stage rough)   (on-feature 8)  (TAD right)  (relation-with-feature 0))
(operation (number 802)  (type turn)  (machining_stage semifinish)   (on-feature 8)(TAD right)(relation-with-feature 0))
(operation (number 803)  (type turn)  (machining_stage finish)   (on-feature 8)  (TAD right)    (relation-with-feature 0))
(operation (number 901)  (type turn)  (machining_stage rough)   (on-feature 9)  (TAD right)  (relation-with-feature 0))
(operation (number 902)  (type turn)  (machining_stage semifinish) (on-feature 9)  (TAD right)(relation-with-feature 0))
(operation (number 903)  (type turn)  (machining_stage finish)   (on-feature 9)  (TAD right) (relation-with-feature 0))
(operation (number 10)  (type grooving)     (on-feature 10) (TAD right)     (relation-with-feature 0))
(operation (number 11101) (type turn)   (machining_stage rough)    (on-feature 111) (TAD right)(relation-with-feature 0))
(operation (number 11102) (type turn) (machining_stage semifinish)(on-feature 111)(TAD right)(relation-with-feature 0))
(operation (number 11103) (type turn)   (machining_stage finish) (on-feature 111) (TAD right) (relation-with-feature 0))
(operation (number 11)  (type threading)     (on-feature 11) (TAD right)     (relation-with-feature 0))
(operation (number 12)  (type grooving)      (on-feature 12) (TAD right)     (relation-with-feature 0))
(operation (number 1301)  (type turn)   (machining_stage rough)    (on-feature 13) (TAD right) (relation-with-feature 0))
(operation (number 1302)  (type turn) (machining_stage semifinish) (on-feature 13) (TAD right) (relation-with-feature 0))
(operation (number 1303)  (type turn)   (machining_stage finish) (on-feature 13) (TAD right) (relation-with-feature 0))
(operation (number 1401)  (type turn-face)  (machining_stage rough) (on-feature 14)(TAD right)(relation-with-feature 0))
(operation (number 1402)(type turn-face)(machining_stage semifinish)(on-feature 14)(TAD right)(relation-with-feature
0))
(operation (number 1403)  (type turn-face) (machining_stage finish) (on-feature 14) (TAD right) (relation-with-feature 0))
(operation (number 1501)  (type drill)    (on-feature 15) (TAD left)      (relation-with-feature 0))
(operation (number 15001)  (type bore)   (machining_stage rough)   (on-feature 15) (TAD left) (relation-with-feature 0))
(operation (number 15002)  (type bore)   (machining_stage semifinish)(on-feature 15)(TAD left) (relation-with-feature 0))
(operation (number 15003)  (type bore)   (machining_stage finish)   (on-feature 15) (TAD left) (relation-with-feature 0))
(operation (number 1601)  (type drill)      (on-feature 16) (TAD left) (relation-with-feature 0))
(operation (number 16001)  (type bore)   (machining_stage rough)   (on-feature 16) (TAD left) (relation-with-feature 0))
(operation (number 16002)  (type bore)  (machining_stage semifinish)(on-feature 16) (TAD left) (relation-with-feature 0))
(operation (number 16003)  (type bore)   (machining_stage finish)   (on-feature 16) (TAD left) (relation-with-feature 0))
(operation (number 17)  (type deep-hole-drill)  (on-feature 17) (TAD right-left) (relation-with-feature 0))
(operation (number 1801)  (type drill)      (on-feature 18) (TAD right)      (relation-with-feature 0))
(operation (number 18001)  (type bore)   (machining_stage rough)   (on-feature 18) (TAD right) (relation-with-feature 0))
(operation (number 18002) (type bore)(machining_stage semifinish)(on-feature 18) (TAD right) (relation-with-feature 0))
(operation (number 18003)  (type bore) (machining_stage finish) (on-feature 18) (TAD right)   (relation-with-feature 0)))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; file name: datafile.clp (for illustrative example 2);;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;***************************
;;;* DEFFACTS KNOWLEDGE BASE *
;;;***************************

(deffacts MAIN::features_list
        (feature (number 1) (name FACE) (type EXTERNAL)  (subtype PRIMARY) (adjacent_features 2 10)
(adjacent_features_names EXTERNAL_STEP HOLE)(TAD left))
        (feature (number 2) (name EXTERNAL_STEP) (type EXTERNAL)  (subtype PRIMARY) (adjacent_features 1 3)
(adjacent_features_names FACE EXTERNAL_TAPER)(step_diameter 45)(TAD right-left))
        (feature (number 3) (name EXTERNAL_TAPER) (type EXTERNAL)  (subtype PRIMARY) (adjacent_features 2 4)
(adjacent_features_names EXTERNAL_STEP EXTERNAL_STEP)(TAD right))
        (feature (number 4) (name EXTERNAL_STEP) (type EXTERNAL)  (subtype PRIMARY) (adjacent_features 3 5)
(adjacent_features_names EXTERNAL_TAPER EXTERNAL_STEP)(step_diameter 22)(adjacent_step_diameters 17)(TAD right))
        (feature (number 5) (name EXTERNAL_STEP) (type EXTERNAL)  (subtype PRIMARY) (adjacent_features 4 6)
(adjacent_features_names EXTERNAL_STEP GROOVE)(step_diameter 17)(adjacent_step_diameters 22)(TAD right))
        (feature (number 6) (name GROOVE) (type EXTERNAL)  (subtype PRIMARY) (adjacent_features 5 7)
(adjacent_features_names EXTERNAL_STEP THREAD)(TAD right))
        (feature (number 70) (name EXTERNAL_STEP) (type EXTERNAL)  (subtype PRIMARY) (adjacent_features 6 8)
(adjacent_features_names GROOVE FACE)(step_diameter 15)(TAD right))
        (feature (number 7) (name THREAD) (type EXTERNAL)  (subtype SECONDARY) (secondary_feature_to
70)(adjacent_features 6 8) (adjacent_features_names GROOVE FACE)(TAD right))

        (feature (number 14) (name KEYWAY) (type EXTERNAL)  (subtype SECONDARY)  (secondary_feature_to
7)(adjacent_features_names FACE)(TAD right))
        (feature (number 8) (name FACE) (type EXTERNAL)  (subtype PRIMARY) (adjacent_features 7 10)
(adjacent_features_names THREAD HOLE)(TAD right))

        (feature (number 9) (name HOLE) (type INTERNAL) (subtype PRIMARY) (hole_diameter 18) (hole_depth 2)
(adjacent_features 1 10) (adjacent_features_names FACE HOLE) (adjacent_hole_diameters 20) (adjacent_hole_depths 3)(TAD left))
        (feature (number 10) (name HOLE) (type INTERNAL)  (subtype PRIMARY) (hole_diameter 20) (hole_depth 3)
(adjacent_features 9 11) (adjacent_features_names HOLE HOLE) (adjacent_hole_diameters 18 17) (adjacent_hole_depths 2 6)(TAD
left))
        (feature (number 11) (name HOLE) (type INTERNAL)  (subtype PRIMARY) (hole_diameter 17) (hole_depth 6)
(adjacent_features 10 12) (adjacent_features_names HOLE HOLE) (adjacent_hole_diameters 20 19) (adjacent_hole_depths 3
7)(TAD left))
        (feature (number 12) (name HOLE) (type INTERNAL)  (subtype PRIMARY) (hole_diameter 19) (hole_depth 7)
(adjacent_features 11 13) (adjacent_features_names HOLE HOLE) (adjacent_hole_diameters 17 10) (adjacent_hole_depths 6
60)(TAD left))
        (feature (number 13) (name HOLE) (type INTERNAL)  (subtype PRIMARY) (hole_diameter 9) (hole_depth 60)
(adjacent_features 12 8) (adjacent_features_names HOLE FACE) (adjacent_hole_diameters 19) (adjacent_hole_depths 7)(TAD
right-left))
        (order-of-features-from-left-external-to-internal (feature-numbers 1 2 3 4 5 6 7 8 9 10 11 12 13 14))
        (order-of-features-from-right-external-to-internal (feature-numbers 14 13 12 11 10 9 8 7 6 5 4 3 2 1))
        (feature-with-largest-dia (number 2))
        )

(deffacts MAIN::machining-operations
        (operation (number 101) (type turn-face)  (machining_stage rough)  (on-feature 1) (TAD left)  (relation-with-feature 0))
        (operation (number 102)(type turn-face) (machining_stage semifinish) (on-feature 1) (TAD left) (relation-with-feature 0))
        (operation (number 103) (type turn-face)  (machining_stage finish) (on-feature 1)  (TAD left) (relation-with-feature 0))
        (operation (number 201)  (type turn)  (machining_stage rough) (on-feature 2)  (TAD right-left) (relation-with-feature 0))
        (operation (number 202) (type turn) (machining_stage semifinish)(on-feature 2)(TAD right-left) (relation-with-feature 0))
        (operation (number 203)(type finish-turn)(machining_stage finish)(on-feature 2)(TAD right-left)(relation-with-feature 0))
        (operation (number 3)  (type turn-taper) (machining_stage rough) (on-feature 3)  (TAD right) (relation-with-feature 0))
        (operation (number 301)  (type turn-contour) (machining_stage rough)(on-feature 3)(TAD right)(relation-with-feature 0))
        (operation (number 302)(type turn-contour)(machining_stage semifinish)(on-feature 3)(TAD right)(relation-with-feature
0))
        (operation (number 303)  (type turn-contour) (machining_stage finish)(on-feature 3)(TAD right)(relation-with-feature 0))
        (operation (number 4)  (type turn-taper) (machining_stage rough) (on-feature 4)  (TAD right)  (relation-with-feature 0))
        (operation (number 401) (type turn-contour)  (machining_stage rough)(on-feature 4) (TAD right)(relation-with-feature 0))
        (operation (number 402)(type turn-contour)(machining_stage semifinish)(on-feature 4)(TAD right)(relation-with-feature
0))
        (operation (number 403) (type turn-contour)  (machining_stage finish)(on-feature 4) (TAD right)(relation-with-feature 0))
        (operation (number 501) (type turn-contour)  (machining_stage rough)(on-feature 5)(TAD right)(relation-with-feature 0))
        (operation (number 502)(type turn-contour)(machining_stage semifinish)(on-feature 5)(TAD right)(relation-with-feature
0))
        (operation (number 503) (type turn-contour) (machining_stage finish)(on-feature 5) (TAD right)(relation-with-feature 0))
        (operation (number 6) (type grooving)    (on-feature 6) (TAD right)    (relation-with-feature 0))
        (operation (number 7001) (type turn)  (machining_stage rough)   (on-feature 70) (TAD right) (relation-with-feature 0))
        (operation (number 7002) (type turn)  (machining_stage semifinish)(on-feature 70) (TAD right) (relation-with-feature 0))
        (operation (number 7003) (type turn)  (machining_stage finish)   (on-feature 70) (TAD right) (relation-with-feature 0))
        (operation (number 7) (type threading)    (on-feature 7) (TAD right) (relation-with-feature 0))
        (operation (number 14) (type mill)  (on-feature 14) (TAD right) (relation-with-feature 0))
        (operation (number 801) (type turn-face) (machining_stage rough) (on-feature 8) (TAD right) (relation-with-feature 0))
        (operation (number 802) (type turn-face) (machining_stage semifinish)(on-feature 8)(TAD right)(relation-with-feature 0))
        (operation (number 803) (type turn-face) (machining_stage finish) (on-feature 8) (TAD right) (relation-with-feature 0))
        (operation (number 901) (type drill)    (on-feature 9) (TAD left)     (relation-with-feature 0))
        (operation (number 9001) (type bore)  (machining_stage rough)  (on-feature 9) (TAD left)     (relation-with-feature 0))
        (operation (number 9002) (type bore)  (machining_stage semifinish)  (on-feature 9) (TAD left) (relation-with-feature 0))
        (operation (number 9003) (type bore)  (machining_stage finish)  (on-feature 9) (TAD left) (relation-with-feature 0))
        (operation (number 1001) (type drill)    (on-feature 10) (TAD left)     (relation-with-feature 0))
        (operation (number 10001) (type bore)  (machining_stage rough)  (on-feature 10) (TAD left) (relation-with-feature 0))
        (operation (number 10002) (type bore) (machining_stage semifinish) (on-feature 10) (TAD left) (relation-with-feature 0))
        (operation (number 10003) (type bore)  (machining_stage finish)  (on-feature 10) (TAD left) (relation-with-feature 0))
        (operation (number 1101) (type drill)    (on-feature 11) (TAD left)     (relation-with-feature 0))
        (operation (number 11001) (type bore)  (machining_stage rough)  (on-feature 11) (TAD left) (relation-with-feature 0))
        (operation (number 11002) (type bore)  (machining_stage semifinish)(on-feature 11)(TAD left) (relation-with-feature 0))
        (operation (number 11003) (type bore)  (machining_stage finish)  (on-feature 11) (TAD left) (relation-with-feature 0))
        (operation (number 1201) (type drill)    (on-feature 12) (TAD left) (relation-with-feature 0))
        (operation (number 12001) (type bore)  (machining_stage rough)  (on-feature 12) (TAD left) (relation-with-feature 0))
        (operation (number 12002)(type bore) (machining_stage semifinish)  (on-feature 12) (TAD left) (relation-with-feature 0))
        (operation (number 12003) (type bore)  (machining_stage finish)  (on-feature 12) (TAD left) (relation-with-feature 0))
        (operation (number 1301) (type drill) (on-feature 13) (TAD right-left) (relation-with-feature 5 9)(tolerance 0.1 0.3))
)
```