A Task-level Robot Planner for Assembly Operations [1]

Rukmini Vijaykumar
Michael A. Arbib

COINS Technical Report 86-31

July 1986

Laboratory for Perceptual Robotics
Department of Computer and Information Science
A305, Graduate Research Center
University of Massachusetts
Amherst, MA 01003

A preliminary version of this paper was presented at the IFAC *Symposium on Robot Control*, 6-8th November, 1985, Barcelona, Spain.

Abstract

We describe our current efforts toward building a high-level planner suitable for robots functioning in the assembly domain. We detail the overall system architecture in which planning is composed of two distinct phases, a *strategic planning* phase which works from a broad definition of the assembly task and description of objects, and a *tactical planning* phase which takes in a restricted set of task-level operations and outputs a series of position and stiffness signals to the lower-level arm and hand controllers. This report decribes the initial design and development for the strategic planner within this framework. Our current implementation and the behavior of the system on a few simple assembly tasks are detailed.

---

[1] Preparation of this paper was supported in part by grant number IST-8513989 from NSF

1. Introduction

The planning problem in the robotic domain can be characterized as one of transforming a set of task specifications provided by the user, which are at the level of identifying what objects are involved in the task to a set of commands executable by a robot control system. The approach adopted by traditional planning systems [11], [28], [36] is based on the specification of an initial world model and the changes effected by each primitive action on the current state of the world. However, the characterization of the initial world and the effects of actions are rigidly specified. In the case of a robot system operating in the assembly domain, the information available to the system is often inaccurate. There are essentially three different sources of uncertainty: uncertainty in the geometric models of objects in the robot's environment; uncertainty in the information obtained from sensors; and uncertainty arising from the inaccuracies of the manipulator control system [3]. The subactions, such as grasp, move and manipulate are parametrized and the choice of parameters for each of these subactions is highly interdependent. Thus, all the information pertaining to the applicability of operators may not be known a priori and it is difficult to predict the effects of actions with certainty. In an effort to contend with dynamic changes in the environment and inherent uncertainty, we have extended the role of planning to include monitoring for the successful execution of the plan, and neccessary replanning as well. In essence, the planner does not function as an off-line planner, but instead deploys the sensors to monitor the environment and correspondingly effects changes in the plan, where necessary. We refer to such an approach to planning as ***dynamic planning.*** A more detailed account of the motivations and objectives for taking such an approach are detailed in [35].

We consider the salient characteristics of a dynamic planning component of a robot sytem to be the following:

1. It ***incrementally*** constructs a plan, starting with ***inadequate*** information and procuring information and ***refining*** the plan as it proceeds.

2. It ***dynamically revises*** its plans and goals corresponding to perceived changes in the environment. In particular, the plan is considered to need revision when the environment resulting from the execution of the plan fails to meet certain ***specified criteria.***

3. Planning and execution may be ***interleaved***; it is not necessary that all the planning precede all the execution.

The problem of decomposing and successfully executing a specified assembly task requires decision making at several levels. In order to separate the global considerations (e.g., ordering of the assembly operations, accessibility of an object, choice of grasp sites on an object) in arriving at a reasonable plan for the given task from more local considerations (e.g., specific configuration of the hand to be used in grasping an object, local obstacle avoidance, choice of appropriate sensors to be invoked to verify the successful execution of a task) we have separated planning into two components:

- The *strategic planner* which, from models of the task, environment and the robot, arrives at the appropriate sequence of operations that can be executed with the aid of sensors and knowledge of a fairly localized segment of the environment;
- The *tactical planner*, which then executes the operations output by the strategic planner with the aid of dynamic sensory information by sending appropriate signals to the robot controller as well as the sensors.

In addition to these two planning subsystems, we have delegated the problem of finding a collision-free path for the robot arm and hand in effecting gross motions to be carried out by a distinct subsytem, the *path planner*. The path planner takes as input the intended start and goal locations of the robot end-effector and outputs a piecewise linear path that avoids collisions with other objects in the environment. The path planner may be invoked by both the strategic and tactical planners. This paper concerns itself with the development of the strategic planner and its interface with the other subsystems.

The planning efforts in the field of robotics itself have centered around the development of high-level languages that would require the user to specify a task in terms of the physical relationships between objects rather than the robot motions required to carry out the task. To date, no complete system has been implemented due to the complexity of the problem. Examples of such systems are AUTOPASS [16] and LAMA [19]. However, considerable progress has been made in specific subproblems of the overall problem of planning robot motions, namely, grasp planning [18], [13], [14], gross motion planning [34], [20], [4], [5], [6] and fine-motion planning [32], [9], [21], [24]. Currently, there are revived efforts to solve the overall planning problem by integrating solutions to the specific subproblems [22], [15]. As the solutions to the subproblems are highly inter-dependent (as in choice of grasp parameters and the subsequent fine-motion that is carried out), these systems propose constraint propagation as a suitable mechanism for dealing with the inter-dependencies. As opposed to this bottom-up approach of combining individual solutions to subproblems,

our work attempts to allow the constraints to evolve as we work on the problem proceeding in a top-down fashion.

The approach we are taking towards building a dynamic planner is to develop a static planner for the assembly domain, with a simple control structure. As a first step to dynamic planning we allow the planner to request simulated sensory input. Subsequently, we will simulate manipulator actions and also experiment with more flexible control structures. The final phase of the work will be to integrate the planner with an actual robot system. The following section describes the overall system of which the planner is a part. Section 3 details the problem domain we are working with, the input language to the strategic planner, the structure of the object database the planner interacts with and the commands ouput to the tactical planner. Section 4 describes the approach we adopt towards problem decomposition and details the components of our current implementation of the planner. In section 5, we discuss the behavior of the planner on three simple assembly tasks that we have studied in some detail. Section 6 discusses limitations of our current system and future directions of research.

2. System Overview

The overall system consists of the strategic planner, tactical planner, path planner, object database, sensory systems, controllers for the arm, wrist and hand and the necessary hardware. A schematic of the overall system and the relationship between the various subsystems is shown in Figure 1.

We start by outlining the basic characteristics of the underlying hardware. We assume:

- a six degree of freedom robot arm which has three positional degrees of freedom, and a three rotational degrees of freedom wrist;
- a dextrous hand mounted on the wrist;
- availability of force information from the wrist and from strain gauges mounted on the fingers of the dextrous hand;
- tactile sensors mounted on the fingers and palm of the hand;
- a visual system that produces location information (position and orientation in three dimensional space), and also some visual features. This would involve a stereo camera system, or a camera and range-finder, connected to a preprocessor system.

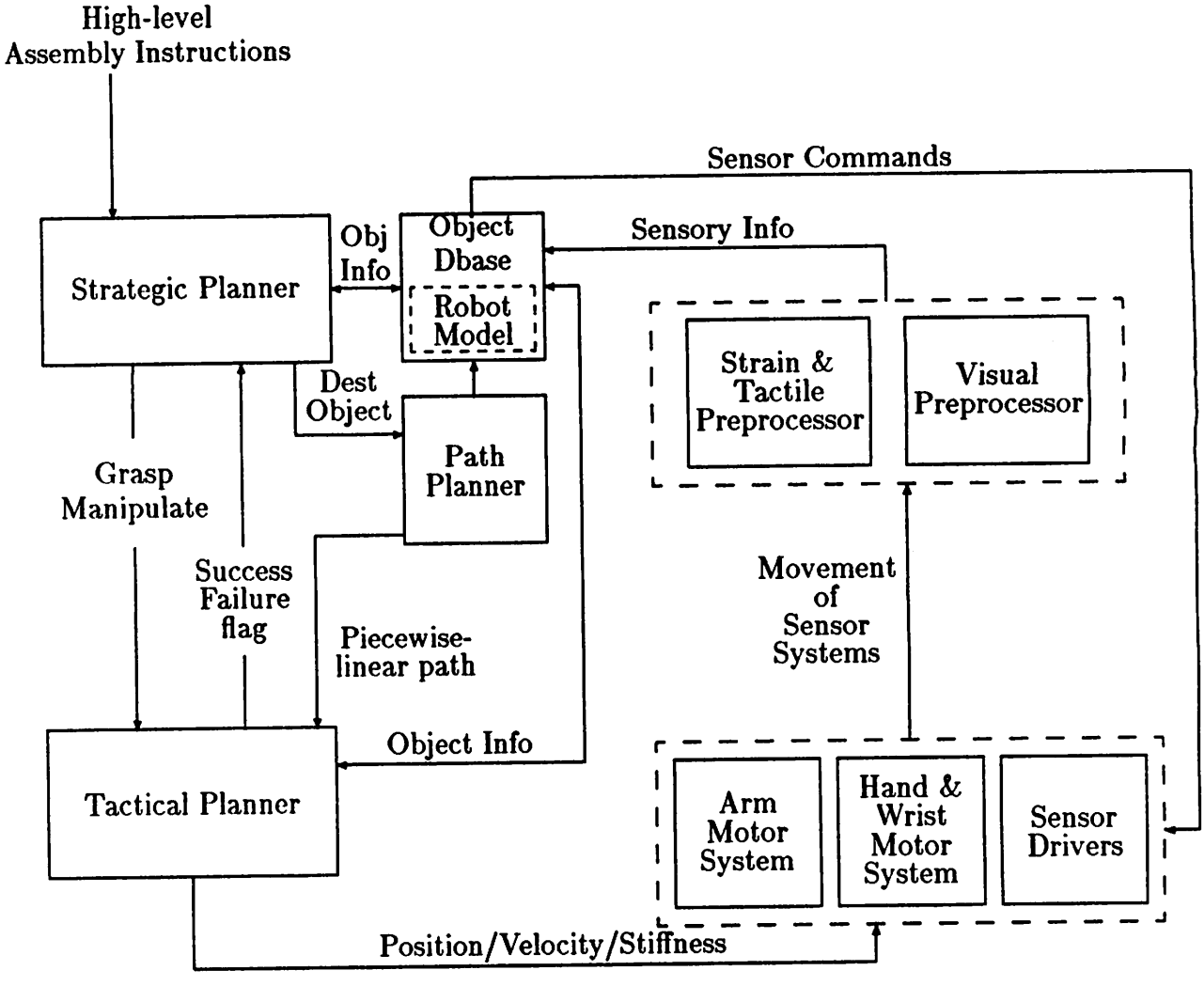

Figure 1: Schematic of the Overall System

Above and separate from the sensor hardware we assume some preprocessing elements. A tactile preprocessing system implements useful tactile algorithms as has been described in [2] and [25]. Tactile and force information will be dealt with in a unified fashion. The visual preprocessor takes the input from the camera system and produces object features. The preprocessor could use information from the object database, allowing it to classify objects.

We separate the control of the robot system at the lowest level into a hand controller, an arm controller, and sensor drivers which effect movement of sensory systems such as moving the camera. Separate controllers are used for arm and hand, because their respective functions are quite different. The arm is used only for gross position placement, fine-tuning being accomplished by the fingers of the hand. The arm controller allows the hand to be directed along a given trajectory. Both the hand and arm controllers share control of the wrist degrees of freedom. The hand controller allows the fingers of the hand to be placed in particular configurations, and also allows the stiffness of the fingers to be controlled.

The high-level robot control is divided into two hierarchical units: the strategic planner, which takes a broad definition of the assembly task and, using the data-base of objects, generates a more specific sequence of operations; and the tactical planner, whose input is a restricted set of task-level operations, and whose output is a series of position and stiffness signals to the lower-level arm and hand controllers. A path planner and object database are subsytems that are used by both strategic and tactical planners. The remainder of this document deals in more detail with the strategic planner component of the system.

3. The Strategic Planner

The problem domain that we are concerned with, initially, is that of simple assembly tasks. The parts lie in the workplace of a robot and the planner is given a very high-level plan for the assembly of the finished product. The high-level instructions will refer to objects by name and will not generally include any specification of parameters for actual robot movements, such as position, velocity, stiffness, etc. These instructions are then broken down into more detailed instructions using world knowledge present in the system and environmental knowledge obtained using sensors. The assembly plan is thus refined at several levels of detail until the plan can be expressed as a set of low-level commands that may be directed at the sensory system or the manipulator control system.

3.1 Input Language to the Strategic Planner

The problem that the strategic planner is faced with is one of starting from a set of initial instructions, which are of the form:

operation *object1 .. object2*

Some examples are:

Fit X into Y
Thread U with V
Double-insert A into B

Starting with a set of instructions of this nature, and a description of the objects present in the workspace in the object database, the strategic planner attempts to break down the given instructions into a series of output commands to the tactical planner.

3.2 Object Database

The object database contains information relating to each of the objects involved in the assembly. As can be seen from Figure 1, the object database is also shared by the tactical planner and is updated by the preprocessed information obtained from the sensors.

We are currently concerned with the class of rigid objects. While we store some general information about an object and its gross dimensions in the representation of an object itself, more detailed information about an object is stored in substructures that describe the features of an object. A *feature* is defined as "a specific geometric configuration formed on the surface, edge, or corner of a workpiece intended to modify or to aid in achieving a given function."[7]. Such a representation for objects has been motivated by the fact that in most assembly operations, it is the specific features of objects that dictate how these objects may be assembled together. The features we consider are quite similar to the ones considered by Popplestone, Ambler and Bellos in describing spatial relationships between features [27].

The information pertaining to each object is stored in a frame structure. We maintain the following information about each object in the workspace:

- general shape,

- length, width, and height (which are derived as the dimensions of the smallest rectangular parallelopiped into which the part will fit. Its largest dimension is expressed as the length of the part, the second largest as its width and the smallest as its height.),
- volume,
- surface area,
- position (expressed in terms of the workspace coordinates),
- a list of features, such as cylindrical hole, thread, etc.

We embed a coordinate frame within each object. The origin of the coordinate frame lies at the center of mass of the object and the X, Y, and Z-axes of the frame are along the length, width and height of the object[2]. Features are specified in terms of the object coordinate frame, circumventing the problem of having to update the position of each feature when the object is moved. Thus, the position slot in an object frame is in fact a 4x4 transformation matrix, specifying the location of the origin and orientation of the axes of the object coordinate frame in terms of the workspace coordinate frame[26].

The data preprocessed and received from various sensors are used to dynamically update the object database. These object models integrate standard structural information about the object along with functional information as to how the object may be recognized, inspected, and manipulated. The strategic planner uses these models for object representation and when features relevant to the planning process are absent in the model, the object database module is responsible for directing the sensory systems to attempt to obtain this information. In addition, the knowledge about objects embodied in the object database can be used as constraints to guide the identification of objects that are of interest to the current planning problem.

3.2.1 Object Features

While we have not restricted the objects handled by the planner in terms of their shape, currently our planner is only equipped to handle the types of features listed in Table 1. The information relating to features fall into two categories: geometric and topological. On the

[2]These choices for origin and axes of the object coordinate frame are a possible set of choices; not necessary ones. The basic scheme works even if a different origin and orientations for axes are chosen.

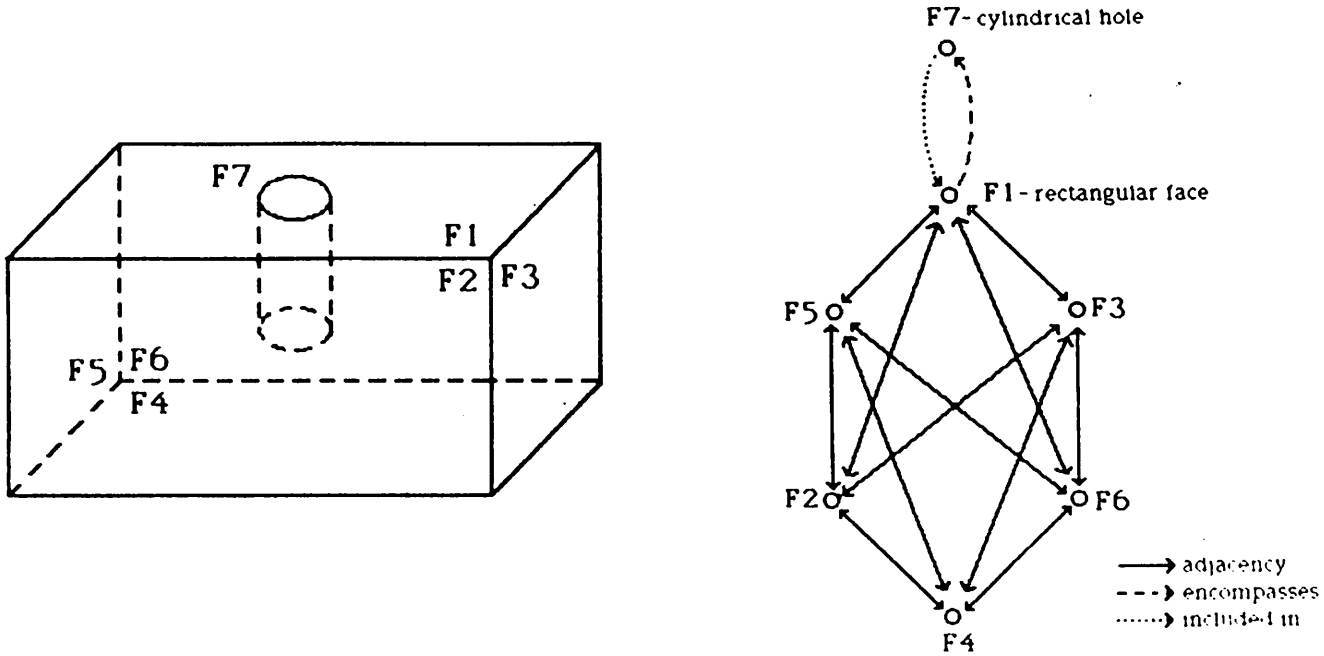

Figure 2: An Object and its Feature Graph

one hand, there is positional and dimensional information required to define the geometry of a feature. On the other hand, there is information pertaining to how the features of a single object or subassembly are positioned with respect to each other. This is expressed as relationships between the features of an object or subassembly. The relationships currently defined are: *adjacent, overlapping, inclusive* and *encompassing.* Conceptually, the features of an object may be represented as a graph, where each feature is represented as a node, and a relationship existing between two features is represented by an arc, appropriately labelled, between the corresponding nodes. We will refer to this entity as the *feature graph* of an object. An example of a box with a hole in it is seen in Figure 2, along with its corresponding feature graph.

While toplogical information can be deduced from the positional information, it is appropriate to represent this explicitly, in order to be able to infer the possible effects of operations involving one feature on related features of the same object or subassembly. For example, in Figure 2, an assembly task operation might require a peg to be *inserted* into the hole (feature F7). Another operation might require another block of a similar size to be *placed* adjacent to any one of the faces (features F1 through F6). Since F7 is included

Feature Type	Dimensional Information
Cylindrical hole:	internal-diameter of the cylinder depth of the hole
Rectangular hole:	length of the rectangle width of the rectangle depth of the hole
Thread:	whether the thread is internal or external diameter width of the thread pitch of the thread
Rectangular plane face:	length of the rectangle width of the rectangle
Circular plane face:	diameter of the circle
Cylindrical surface:	height of the cylinder diameter of the cylinder
Semi-cylindrical depression:	diameter depth

Table 1: Feature Types and Dimensional Information

in F1, their relationship constrains the *place* operation to be relative to only features F2 through F6.

Individual features are described by specifying their position, dimensions and type (cylindrical hole, rectangular hole, etc.). The position of each feature is described by specifying the *center* of the feature and its *normal axis*. The definitions of center and normal axis for each feature type listed in Table 1 are provided in Figure 3. The normal axis always passes through the center of the feature and the direction of the normal axis is away from the object. Thus, the position of a feature may equivalently be described by the specification of the center and the plane passing through the center, perpendicular to the normal axis, expressed in terms of the object coordinate frame.

The dimensional information required about features varies depending on the feature type and is described in Table 1 for each of the feature types.

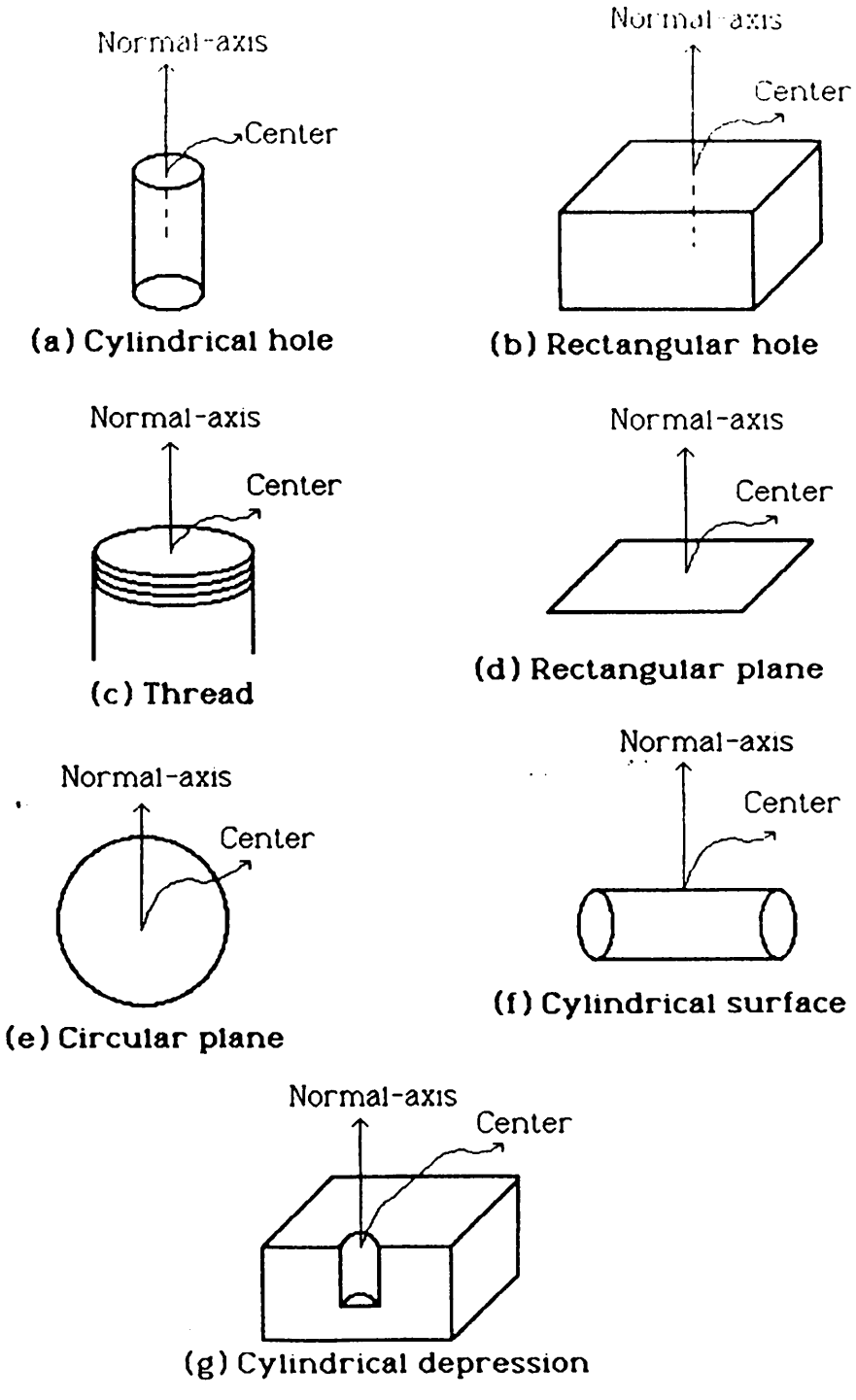

Figure 3: Definition of Center and Normal Axis for Feature types

3.3 Output Commands

As noted in Figure 1, we group the low-level commands produced by the strategic planner into three classes: grasp commands, reorientation commands and manipulation commands. In the following, we list our current working set of commands organized within the three classes:

Grasp Command: As detailed in [1] the grasp action is essentially carried out in two phases: the preshape phase, where the hand is configured to the preshape configuration and carried to the vicinity of the object and the grip phase where the hand acquires the object in a manner suitable for subsequent manipulation.

Grasp*(object, compliance-measure, operation)*: The command instructs the tactical planner to grasp the specified object by providing a set of grasp parameters along with the object identification. A more detailed description of the command parameters are given below:

object: The object parameter corresponds to a structure that includes the following information: the feature(s) to be used for grasping specified by their position relative to the workspace coordinates; the normal axis (or axes) of the mating feature(s) of the object being grasped.

compliance-measure: This parameter indicates the extent of compliance required in the subsequent operation. The possible range of values lie between 0 and 10, where a specification of 10 would imply a need for maximum compliance and a specification of 0 would imply a firm grasp.

operation: This parameter specifies the assembly operation to be executed in order that the grasping subsystem may choose an appropriate grasp configuration.

The feedback from the tactical planner to the strategic planner is a signal indicating whether the Grasp command was successfully executed. In case of failure, it would indicate whether the failure occurred in the preshape or grip phase of the operation. A failure in the preshape phase would indicate that due to the presence of other objects in the environment the grasp configuration chosen, and consequently the preshape configuration, is inappropriate. A failure in the grip phase may arise either from a failure to acquire the object stably, or because the resulting grasp configuration overconstrains the object for the subsequent manipulation.

Reorientation Commands:

All of the following commands assume that the robot hand is currently holding an object. They are expressed in terms of a relation that is required to exist between a point, line or plane in the object coordinate frame and a corresponding entity in the workspace coordinates. Axes and planes are oriented; each axis has a specified direction (so that two collinear axes may be aligned or opposed), and each plane has a specified normal. These reorientation commands are not directly passed on to the tactical planner, but instead are sent to the path planner and the trajectory produced by the path planner is then passed on to the tactical planner.

oppose*(axis1, axis2)*: *axis1* is specified in terms of the coordinate frame of the object being held, while *axis2* is specified in terms of the workspace coordinates. The command instructs the robot to reorient the object in such a manner that the two specified axes are opposed (i.e., the axes lie along the same line oriented in opposite directions).

align*(axis1, axis2)*: *axis1* and *axis2* are to coincide, with their directions being the same.

place-parallel*(axis1, axis2, [plane])*: *axis1* and *axis2* are to be made parallel. When an optional parameter specifying a plane is provided, there is an added requirement that *axis1* should lie on the specified plane.

place-perpendicular*(axis1, axis2, [plane])*: *axis1* is to be made perpendicular to *axis2*. When a plane is also specified, *axis1* is required to lie on the specified plane.

place*(point1, point2)*: This command requires the robot hand to reorient the object being held such that *point1* specified in terms of the object coordinate frame coincides with *point2* specified in the workspace coordinate frame.

align*(point1, line)*: This command requires that *point1*, which is specified in the object coordinate frame, lie on the specified line. The parameter *line* is specified in the workspace coordinate frame.

coplanar*(axis1, plane)*: *axis1* is to be made to lie on the specified *plane*.

merge*(plane1, plane2)*: This command requires a reorientation of the object such that *plane1*, specified in terms of object coordinates, coincides with *plane2*, specified in terms of workspace coordinates.

against*(plane1, plane2)*: Requires that the normals of the planes be opposed while the planes coincide.

The path planner, when successful in finding a collision-free path to achieve the specified pose, passes this path to the tactical planner. When it fails to find a path, failure indication is sent back to the strategic planner along with the identification of objects in the workspace that act as obstacles for possible use in plan modification. There is still a possibility that the tactical planner may succeed or fail in accomplishing the pose. The feedback from the tactical planner on all of these commands is an indication as to whether the operation was successfully executed, the part was dropped, or if there was a failure to achieve the required pose. In the case of failing to achieve the specified position and orientation while still holding the object, the tactical planner passes back to the strategic planner the current pose of the object and the path segment that is left unexecuted.

Manipulation Commands: These commands assume the object is being held by the hand.

insert*(insertion-axis, object-compliance, insertion-amount)*: Requests the tactical planner to perform an insert action which translates into a translation motion by the specified amount along the *insertion-axis*. The *object-compliance* is specified as a 3-component vector which specifies the compliance along the insertion-axis and two axes orthogonal to the insertion-axis.

thread*(rotation-axis, object-compliance, rotation-amount)*: Requires the tactical planner to execute a thread operation which in turn translates into a rotational motion by the specified *rotation-amount* about the *rotation-axis*. The *object-compliance* parameter is the same as described in the **insert** operation.

double-insert*(insertion-axes, object-compliance, insertion-amount)*: Requires the tactical planner to execute a double-insert operation which translates into a translation motion by the specified *insertion-amount* along the *insertion axes*.

The feedback from the tactical planner on any of the manipulation commands is an indication of whether the execution of the operation succeeded or failed. In case of failure of insert or double-insert operations, the tactical planner returns the actual insertion amount along with the failure flag. In the case of failure of thread operations, the tactical planner returns the amount of rotation achieved, translation of the object along the axis of rotation and the termination torque.

4. Structure of the Planner

The input to the planner is provided in terms of specifying the assembly operation to be carried out and the objects involved in the operation. As noted in Figure 4, the planner must refine this task level description into a series of commands to the tactical planner and path planner. The first level of problem refinement is in identifying the specific features of the objects that are to be mated during the assembly operation. Having obtained a feature-level description of the task, knowledge about the specific operations is employed to refine the problem further into a set of spatial relationships that are to hold between the features. The last level of problem refinement is in identifying the tactical planner level commands that would accomplish these spatial relationships. As indicated in Figure 4, problems encountered at any of the levels will percolate to change decisions made at higher levels in the hierarchy. The current implementation adequately handles the first-level of refinement from object to feature-level descriptions. The refinement to spatial relationships is included in the current implementation; however, reasoning about these spatial relationships and generating commands to be communicated to the tactical and path planners are to be implemented in the near future. In the following, we describe the control structure of our current planner, representations used for goals and operators, heuristics used, constraint propagation mechanism and the set of spatial relationships we are concerned with.

4.1 Control Structure

The object information given to the planner provides a partial description of the world model while the high-level assembly instructions provide the goals. The first level of refinement consists in identifying precisely the object features involved in the assembly and subsequent levels to define the goal in terms of spatial relations and primitives to the path and tactical planners. Thus, the goals are refined successively to obtain a clearer statement of the goals, and in subsequent levels, to arrive at a set of subgoals that are easier to solve and whose solutions are sufficient to solve the original problem. Thus, the approach adopted by the planner is one of problem reduction or backward-reasoning while using object related information as heuristics to prune the search tree.

The planner includes explicit representation of the goals, at the various levels described above, as a network of nodes. The network of goals, at any time, reflects the current state of problem refinement. Aside from the goals, the planner includes representation

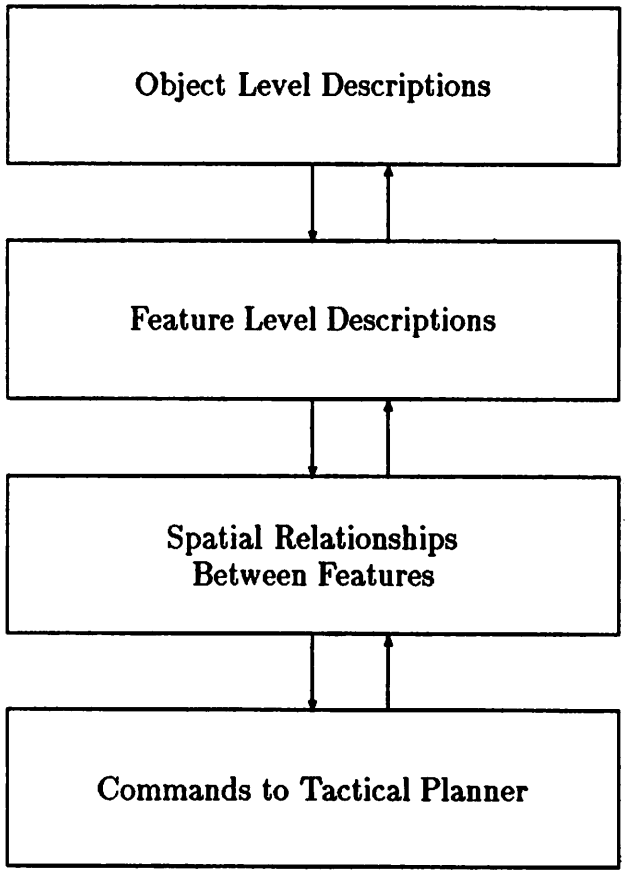

Figure 4: Levels of Problem Refinement

of the operations and the geometric constraints associated with each of the operations; These geometric constraints are used initially to guide the problem refinement process by eliminating possible pairs of mating features that do not satisfy these geometric constraints. At this stage, the planner has no reason to prefer one alternative over the other. This is reflected by associating an equal confidence measure, a number ranging between 0 and 1, with each of the feature level subgoals. A set of reasoning heuristics, specific to assembly operations, are then applied which alter the confidence rates associated with the feature level subgoals appropriately when there is evidence to believe that one set of mating features is more plausible than others. These reasoning heuristics which serve to further refine the problem are represented as rules in a rulebase. When alternative solutions are recognized to be equivalent, the planner selects one and stores the others as *choice points* [31] for later consideration. When the constraints and heuristics do not completely determine a single alternative as the viable one, the planner uses a strategy of *guessing* by picking one that appears most likely to succeed. The planner also includes a mechanism by which accessibility constraints are propagated across the network of goal nodes and analyzed for interactions between the proposed solutions. When such interactions are detected, the choice-point mechanism allows the planner to consider other alternatives. The constraint formulation, propagation and satisfaction mechanisms are also encoded as rules in the rule base. In the following subsections, we describe each of the major components of the planner in more detail.

4.2 Representation of Goals

The input instructions are object-level goals, represented as a network of nodes. As each input instruction is further refined, subgoals are added to the network, maintained at levels which correspond to the levels of problem refinement. The representation resembles the procedural networks employed by Sacerdoti in NOAH [28] and extended by Tate in NONLIN [31]. The specific information associated with each goal as slots in the structure defining a goal is described below:

Goal description: The input instruction, specifying the operation and the objects (or features) involved.

Level: An indication of whether the goal being described is at the *object level*, *feature level*, *spatial relationship level*, or *primitive actions level*.

Node-type: Indicates if the goal being represented is an AND type node (all of the subgoals have to be accomplished) or an OR type node (exactly one of the subgoals has to be accomplished).

Parent-goal: Specifies the goal at the immediately higher level of which the current goal is a subgoal.

Preceding goals: Goals that have been specified or identified to be required to have been completed before initiating execution of the current goal.

Successor goals: Goals that have been specified or identified to be required to execute after the completion of the current goal.

Subgoals: Specifies a list of goals that constitute the set of subgoals of the current goal.

Preconditions: Conditions that have to be true in order that the operation can be executed.

Postconditions: Conditions that are required to be true after execution of the operation. Typically, this is composed of preconditions of successor goals that are likely to be affected by execution of the operation, from geometric considerations.

Effects: Conditions that are made true as a result of accomplishing the current goal.

Choice Points: Alternative subgoals, that could be considered should the current choice fail, are stored in a list.

Heuristics applied: The heuristics that have been used so far for problem solution.

Rules applied: The rules that have been used so far.

The description of a goal as given above blurs the distinction between *goals*, which are conditions that are desired to be true in the world model, and *actions*, which represent operations that effect changes in the world model. Specifically, the preconditions, postconditions and effects slots refer to the action associated with a goal, while the remaining slots refer to the desired state of the world model.

4.3 Representation of Operators

The planner currently handles three operations: FIT, THREAD and DOUBLE-INSERT as seen in Figure 5. With each of these operations, we associate a structure which represents the basic high-level information required to execute the operation. Thus, the structure associated with the FIT operation consists of slots for the specific mating features of the two objects that are involved in the operation and the symmetry of the insertion object about the axis of insertion. In refining an object-level goal of the form "Fit X into Y", the planner instantiates a structure associated with the FIT operation and overlays this structure atop the goal structure described earlier for each of the feature-level subgoals generated. The planner needs to apply some reasoning using heuristics and constraints to determine the specific values to fill the slots before further refinement of the operation can be undertaken. The slots of the structures associated with the operations FIT, THREAD and DOUBLE-INSERT are given in Figure 5.

The angle of rotation slot refers to the maximum amount (in degrees) by which the insertion object has to be rotated, after the normal axes of the two features are aligned and before the movement along the axis takes place. This angle of rotation is derived from considering the symmetry of the mating features identified, about their respective normal axes. For example, inserting a cylindrical peg into a cylindrical hole would have an angle of rotation of 0 degrees as both mating features are completely symmetric about their normal axes. On the other hand, inserting a square peg into a square hole would have an angle of rotation of 90 degrees as each of the mating features has an angle of symmetry of 90 degrees about its normal axis.

Aside from these structures that are used for representing the various options, a set of constraints are associated with each of the operations which are detailed below:

Constraints with the FIT operation:

1. The dimensions of the mating feature of the receiver object should be at least as large as the mating feature of the insertion object.

2. The mating feature of the receiver object should be hollow.

Constraints with the THREAD operation:

1. The mating features of both objects should be of type 'thread'; The threading on

FIT

name of operation:
mating-feature of receiver:
mating-feature of insertion:
angle of rotation:

THREAD:

name of operation:
mating-feature of outer object:
mating-feature of inner object:

DOUBLE-INSERT:

name of operation:
mating-feature-1 of receiver:
mating-feature-1 of insertion:
mating-feature-2 of receiver:
mating-feature-2 of insertion:
distance between mating-features of receiver:
distance between mating-features of insertion:

Figure 5: Operation Structures

the outer object should be internal while the threading on the inner object should be external.

2. The diameter of the cylindrical surface on which the threads lie should be nearly equal.

3. The width and pitch of the threads on either feature should be nearly equal.

Constraints with the DOUBLE-INSERT operation:

1. The distance between the two mating features of the receiver involved in the operation should be nearly equal to the distance between the two mating features of the insertion involved in the double-insert operation.

2. The two features of the receiver should both be hollow.

3. The dimensions of the mating features of the receiver object should be at least as large as the corresponding mating features of the insertion object.

4.4 Use of Confidence Rates

The initial assignment of confidence rates to the various feature-level subgoals is given by 1 over the number of possible alternatives. Any subsequent modifications to the confidence rates associated with feature-level subgoals of the same object level goal are effected so as to maintain the sum of the confidence rates over the feature-level subgoals to be equal to 1. The mechanism we use for altering the confidence rates is as follows:

Let $fg_1, fg_2, ..., fg_n$ denote the feature-level subgoals associated with a single object-level goal. Let $ic_1, ic_2, ..., ic_n$ denote their corresponding confidence rates, before application of a specific heuristic. Let *Promoted-goals* denote the set of subgoals which are favored by the heuristic and *Demoted-goals* denote the remaining set of subgoals. Let $fc_1, fc_2, ..., fc_n$ denote the confidence rates of the feature-level subgoals as a result of the application of the heuristic.

If $fg_i \in$ *Demoted-goals*, then $fc_i = ic_i \div 2$.

For subgoals in *Promoted-goals*, the confidence rates are incremented by an equal amount ϵ.

$fc_i = ic_i + \epsilon$ for $fg_i \in$ *Promoted-goals*

where ϵ is determined as

$$\epsilon = \sum \{ (fc_j - ic_j)/ \mid \textit{Promoted-goals} \mid \quad \Big| \; j, fg_j \in \textit{Demoted-goals} \; \}$$

4.5 Reasoning Heuristics

The planner currently includes three heuristics, all of which are involved with reasoning about the mating features of an operation. These are:

Closeness-of-fit: This heuristic is applicable specifically to goals that represent insertion operations, such as FIT and DOUBLE-INSERT. It essentially compares the difference in dimension between each pair of mating features occuring in each alternative feature-level subgoal associated with a goal, and results in increasing the confidence rate associated with the feature-level subgoals that have the least difference, and correspondingly reducing the confidence rate associated with the remaining subgoals, as described in the previous section.

Distinguishing features: This heuristic works on the feature-level subgoal that has the highest confidence rating. Thus, one may view this as a heuristic that is used to reinforce an earlier generated hypothesis. The heuristic works by requesting more detailed information about the objects involved in the operation around the mating features represented in the specific feature-level subgoal. This would invoke the sensory systems or the object database to obtain this information. The result of this request might be an augmentation to the feature graph, either in terms of identifying new features that are related to the mating feature(s) in consideration, or in terms of identifying new relationships between that mating feature and other existing features. If no change results in the feature graph, the application of the heuristic leaves the solution state intact. If indeed there are changes in the feature graphs of the two objects, the changes may be compared and analyzed to see if there is additional evidence to increase the confidence measure of the feature-level subgoal.

Feature Symmetry: If after applying the above heuristics, there are several possible sets of mating features that can be considered for executing an operation, the feature symmetry heuristic analyzes the options to see if a pair of subgoals are identical except for considering two distinct but symmetric features from the same object.

Two features from the same object are considered to be symmetric if they are of the same type, have the same dimensions and bear the same relationships with other features of the object[3]. When sugoals are found to represent such alternatives, the planner defers one of the subgoals by removing it from the list of subgoals and storing it among the list of choice points for later consideration.

4.6 Constraint formulation and propagation

The information currently encoded as constraints in the system are in terms of the accessibility of features identified to be mating features of an operation. When a feature f_i of an object occurs as a mating feature in a feature-level subgoal, the predicate *accessible* f_i is added to the set of preconditions associated with the corresponding object-level goal.

Subsequently, the constraint is propagated forward to any of the preceding goals that:

- has the object(O) of which f_i is a feature as one its operands;
- has a feature f_j among the mating features of O considered in its list of feature-level subgoals, such that f_i and f_j are related by any one of the relations defined over the set of features.

Any goal that satisfies the above conditions has the constraint added to its set of postconditions.

The propagated constraints may then be used to constrain the solution space because the choice of feature-level subgoal considered for a goal should ensure the satisfaction of these conditions upon completion of the operation. Typically, this would require that the planner build a model of the subassembly resulting from the operation and verify that there is enough clearance to reach the object(O) along the normal axis of the feature (f_i). As a first approximation to this, the planner deduces the inaccessibility of a feature based on the operation, feature type and dimensions of the feature and posts this information among the effects of executing the goal. Examples of this are a close fit or a threading opertion, where the mating features of the operation essentially disappear as a result of executing the operation. For a more formal treatment of this see [17].

[3]We would like to note here that *symmetry* as defined here does not refer to strict positional symmetry but can best be described as functional symmetry

When it is not feasible to deduce conclusively whether or not the execution of an operation violates a postcondition associated with the goal, we handle this by querying the user about whether or not the accessibility constraint will be satisfied, given the operation and mating features. Such a user query may correspond to having the system initiate execution, but with preplanned alternatives, and continue with the execution or backtrack to an assigned point based on sensory information obtained. Such scenarios provide a testing ground for our ideas on dynamic planning where a choice exists between adopting a computationally intensive modelling approach versus an execute-and-replan dynamic planning approach which may have strong demands on real-time response time.

4.7 Transformations from Feature-level to Spatial-Relationships

Refinement of feature-level goals to goals that describe spatial relationships between features is essentially a function of the assembly operation involved. Suppose we have a goal 'Fit $object_1$ into $object_2$' and that the mating features of the objects that have been identified are referred to as f_1 and f_2, respectively. Let α denote the angle of rotation. In refining the problem, two alternatives exist. Either $object_2$ may be positioned stably (using a fixture if necessary), and $object_1$ moved to where $object_2$ is positioned and the operation carried out, or the roles of $object_2$ and $object_1$ may be reversed. We describe below the component actions that are required for one of the alternatives. (The other alternative may be expanded in a similar fashion.)

1. Stabilize $object_2$ (mandatory).
2. Orient $object_2$ so that the normal axis of f_2 is parallel to the Z-axis of the workspace. (this is desirable but not mandatory; i.e., if the normal axis of f_2 is constrained to be in a different direction, this subgoal can be relaxed).
3. Orient $object_1$ so that the normal axis of f_1 is opposed to the normal-axis of f_2 (mandatory).
4. Move $object_1$ along the normal axis of f_1 until the center of f_1 is below the center of f_2 by a distance equal to the depth of f_2.

When α is other than 0, there is an additional step involving rotation of $object_1$ about f_1's normal axis, between steps 3 and 4, which may require that certain other features of the objects bear a specific relationship to each other. For example, in Huber's first example

(refer to Figure 6) the *fitting* operation of part-B into part-A requires a maximum rotation of 180 degrees in order to align the two semi-cylindrical depressions in part-A with the corresponding ones in part-B.

Similarly, let us suppose we have a THREAD task to be accomplished, namely, 'Thread $object_1$ with $object_2$'. The parameter $object_1$ is the enclosing object and $object_2$ is the enclosed object. Let f_1 and f_2 refer to their mating features, respectively. As in the FIT operation, a choice exists as to which one of objects $object_1$ and $object_2$ will be stationary while the other is transported for the mating operation. For the purpose of this section, it suffices to expand on one of the alternatives. The following are the subactions that are required to be executed in order to complete the task:

1. Stabilize $object_2$ (mandatory).
2. Orient $object_2$ so that the normal axis of f_2 is parallel to the Z-axis of the workspace (desirable but not mandatory).
3. Orient $object_1$ so that the normal axis of f_1 is opposed to the normal axis of f_2.
4. Move $object_1$ until $object_1$ and $object_2$ are in contact.
5. Rotate $object_1$ about the merged normal axes until threading is complete (amount of rotation is based on the width and pitch of the threads) and would be monitored with force feedback.

Consider a DOUBLE-INSERT task of the form 'Double-Insert $object_1$ into $object_2$' where the mating features of the receiver ($object_2$) have been identified to be f_1 and f_2 and the mating features of the insertion ($object_1$) are f_1' and f_2', respectively. The feature-level subgoal may be further refined into the following spatial relationships:

1. Stabilize $object_2$ (mandatory).
2. Orient $object_2$ so that the normal axes of f_1 and f_2 are parallel to the Z-axis of the workspace (desirable but not mandatory).
3. Orient $object_1$ so that the normal-axis of f_1' is opposed to the normal-axis of f_1 and the normal-axis of f_2' is opposed to the normal-axis of f_2.
4. Move $object_1$ along the normal-axes of f_1' and f_2' until the centers of f_1' and f_2' are below the centers of f_1 and f_2 by distances equal to the minimum of the depths of f_1 and f_2.

Spatial Relations
opposed(axis1, axis2)
aligned(axis1, axis2)
parallel(axis1, axis2)
coincident(point1, point2)
colinear(point1, point2, line)
coplanar(plane1, plane2)
against(plane1, plane2)
engaged(thread1, thread2)

Table 2: Spatial Relationship Predicates

This suggests that operations at the high level may be broken down into a series of subgoals expressed as predicates, which in turn have to be refined into low-level commands. It seems appropriate to express these intermediate subgoals in terms of spatial relationships between the features of the objects. Our current set of predicates is listed in Table 2. Our current implementation does not reason with these spatial relationships. We intend to use the RAPT system[27] for this purpose. The refinement of these spatial-relationship goals to the corresponding reorientation commands to the path planner, and the generation of appropriate grasp and fine-motion commands are fairly straightforward. However, analyzing the interactions between these commands and choice of appropriate parameters to be used in the grasp and fine-motion commands needs to be worked out.

5. Behavior of the Planner on some Example Assembly Tasks

To illustrate how the current implementation of the planner functions, we will discuss its behavior on three simple assembly tasks. The first example is to assemble a light bulb fixture. The example is due to R.Huber and we will refer to it as Huber's First Example. There are four objects involved in the assembly consisting of three subtasks. The challenging aspect of this task has been the complexity of representing the objects, their features and relationships. The second example is one that involves three objects and two insertion tasks. This example is simple but poses a challenge in terms of dealing with symmetric features and interactions between solutions obtained for the individual object-level goals. The last example has been taken from [33]. The task involves four objects and three double-insert operations. This example poses a combinatorial problem due to the large number of features in one of the objects, in addition to having several symmetric

features in three of the four parts. Computer traces indicating the planner's behavior on these examples are presented in Appendix D.

5.1 Example 1: Huber's First Example

The objects and their features are shown in Figure 6. A complete description of the objects and features are given in Appendix A.

The initial set of instructions given to the planner are as follows:

(G-1): Fit B in A

(G-2): Thread C with A

(G-3): Thread D with B

5.1.1 Program Behavior

Initially, the goals consist of the three given instructions in their specified order. The planner generates goal nodes corresponding to the three given object-level tasks and sets up the precedence relations. The next task for the planner is to refine these goals into feature-level goals. After applying the constraints associated with the operations specified in the respective goals, we are left with three feature-level subgoals for the fit operation (G-1), namely A3-B1, A3-B2 and A3-B4, (the operation constraints reduce the possibilities from 16 to 3); one feature-level subgoal for goal (G-2) (a reduction from 12 possibilities) and two feature-level subgoals for goal (G-3).

The planner then applies the heuristics described earlier in an effort to identify which are the most promising options. The closeness-of-fit heuristic applies to the fit operation and the find-distinguishing-feature heuristic applies to both goals (G-1) and (G-3). Both heuristics apply to the fit operation and result in raising the confidence rate of the A3-B1 subgoal to 0.83 while reducing the confidence measure of the other two to 0.083 each. The distinguishing feature heuristic identifies the flanges and grooves on features A3 and B1. Applied to the goal 'Thread D with B', the same heuristic also identifies the obtruding lip on feature D1; however it does not serve to alter the confidence measures of the subgoals of that operation.

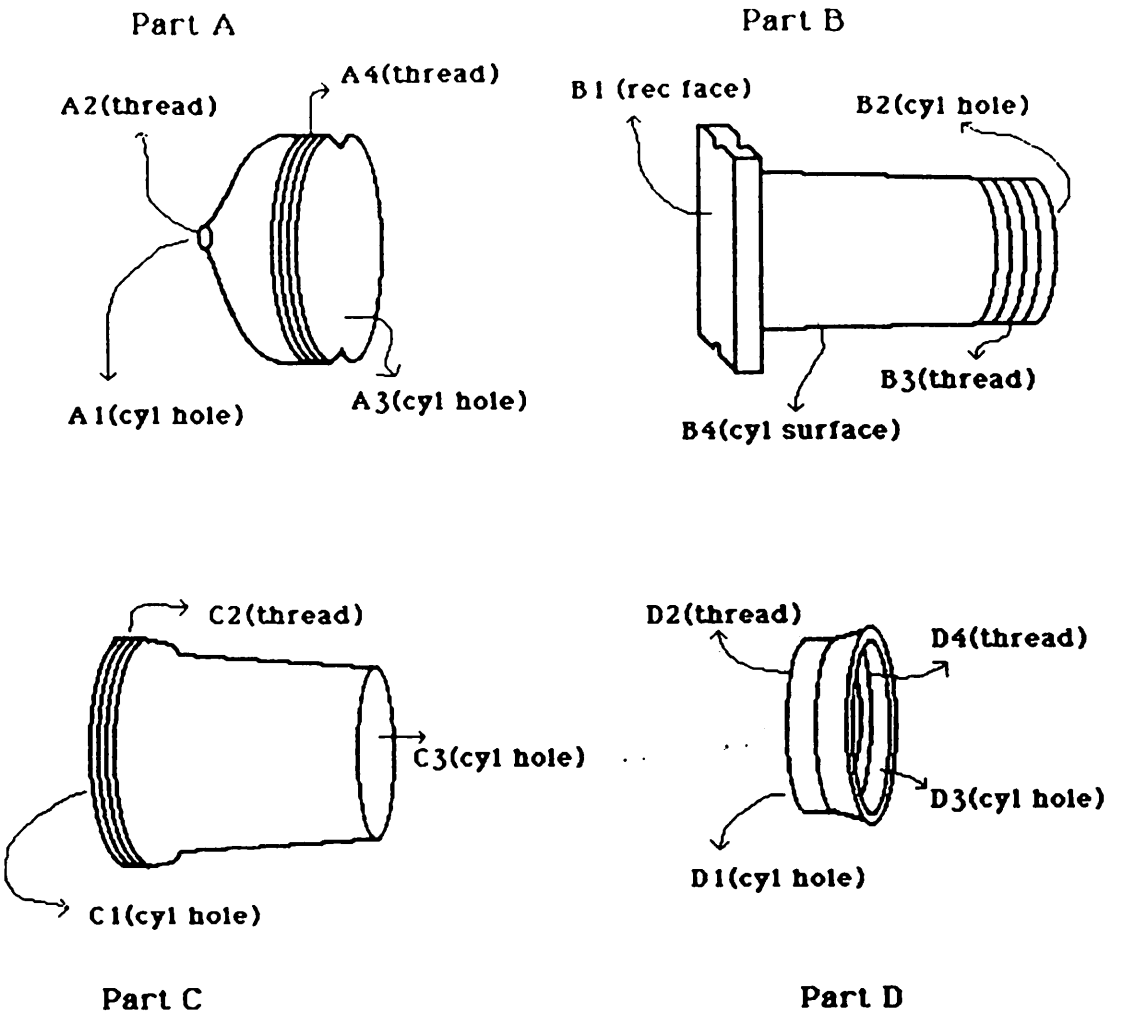

Figure 6: Huber's First Example

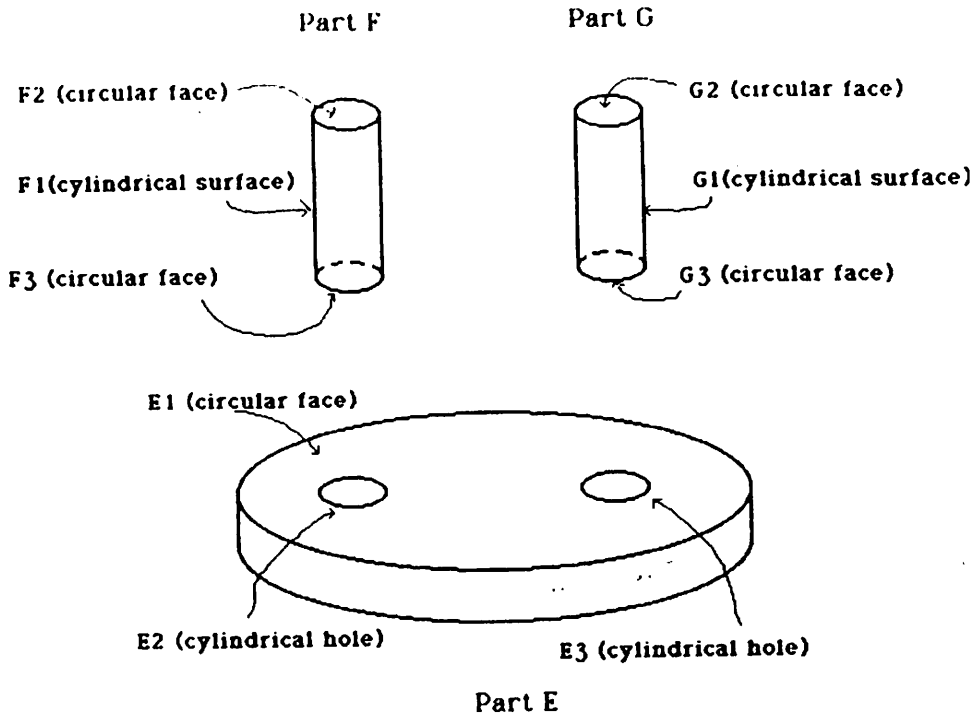

Figure 7: Arbib's Example

Before applying the constraint propagation rules, the planner chooses the one feature-level goal for each of the object-level goals that has the highest confidence rating and defers the other alternatives by storing them in the choice-points slot. The choices made in this example are A3-B1, A4-C2 and B3-D4, respectively. The planner, through the process of constraint propagation, ensures that there is no conflict in the choices made, at the feature-level, and this then is the plan that is executed.

5.2 Example 2: Arbib's Example

This example consists of three objects: a circular plate with two holes and two identical pegs which are to be inserted into the holes. The objects and their features are shown in Figure 7. A complete description of the objects and their features are shown in Appendix B.

The instructions given to the planner are:

(G-1): Fit F into E

(G–2): Fit G into E

5.2.1 Program Behavior

The planner sets up the object-level goals from the given input instructions. The ordering between the goals is assumed to be as given. The refinement process yields four feature-level subgoals for each of the goals (G-1) and (G-2), namely, E2-F2, E3-F2, E2-F3 and E3-F3 for (G-1), and E2-G2, E3-G2, E2-G3 and E3-G3 for (G-2). The application of closeness-of-fit heuristic does not yield any change in the problem state as the features in the various subgoals are of identical dimensions. The feature symmetry rule then identifies that these subgoals involve features that are symmetric and defers all but one of the feature-subgoals of each of the two goals, by adding these subgoals to the list of choice-points. Thus, as a result of applying the heuristics we now have feature-level subgoals corresponding to choice of mating features E3-F3 and E3-G3 corresponding to the two given object-level goals. The constraint propagation phase recognizes that E3-F3 and E3-G3 are conflicting as the use of the feature E3 in the first operation would make it inaccessible to the subsequent operation. The choice E3-F3 is therefore discarded and the planner picks E2-F3 as a potential candidate from among the choice-points available. The constraint propagation rules are then reapplied and this time no conflicts are detected.

5.3 Example 3: Torras's Example

The example consists of four objects. One is a rectangular plate that has six circular holes, five of the same dimension and one that is larger. The other three objects each need to be inserted into the holes in the plate using the double-insert operation. The parts and their features are shown in Figure 8. A complete description of the objects and their features is given in Appendix C.

The instructions given to the planner are:

(G–1): Double-insert I into H

(G–2): Double-insert J into H

(G–3): Double-insert K into H

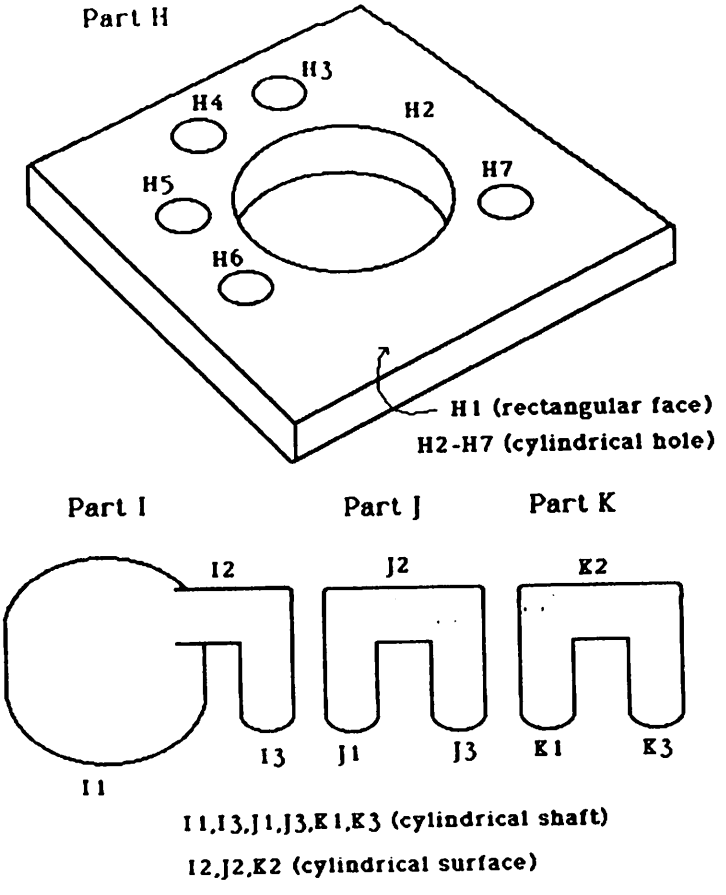

Figure 8: Torras's Example

5.3.1 Program Behavior

The planner initially sets up three object level goals assuming the order of execution of the goals as given. The refinement process to feature-level subgoals yields five distinct possibilities for the first object-level goal "Double-Insert I into H" namely,

(H2-I1; H3-I3), (H2-I1; H4-I3), (H2-I1; H5-I3),
(H2-I1; H6-I3) and (H2-I1; H7-I3).

The other two object-level goals have sixteen feature-level subgoals each. The feature-level subgoals corresponding to (G-2) are:

(H2-J1; H3-J3), (H2-J3; H3-J1), (H2-J1; H4-J3), (H2-J3; H4-J1),
(H2-J1; H5-J3), (H2-J3; H5-J1), (H2-J1; H6-J3), (H2-J3; H6-J1),
(H2-J1; H7-J3), (H2-J3; H7-J1), (H3-J1; H4-J3), (H3-J3; H4-J1),
(H4-J1; H5-J3), (H4-J3; H5-J1), (H5-J1; H6-J3), (H6-J3; H5-J1).

The feature-level subgoals of (G-3) are identical to those resulting from (G-2).

Applying the closeness-of-fit heuristic leaves the confidence measures associated with the feature-level subgoals of (G-1) unchanged, while for the remaining two, the confidence measures of the ten feature-level subgoals involving H2 are reduced and those of the other six are increased. The distinguishing-feature heuristic leaves the states of the subproblems unchanged. The feature-symmetry heuristic recognizes all of the five subgoals of (G-1) to be symmetric and therefore, moves all but the (H2-I1; H7-I3) subgoal to the list of choice-points for later consideration. With (G-2), there are two classes of symmetry, the subgoals that involve H2 fall into one class, and those that do not involve H2 fall into the other. Consequently, from applying the feature-symmetry heuristic this object-level goal has two feature-level subgoals: (H2-J3; H7-J1) and (H5-J3; H6-J1) and the remaining subgoals are stored in the list of choice-points. Similarly, (G-3) has two subgoals: (H2-K3; H7-K1) and (H5-K3; H6-K1).

At this point, the make-guess rule picks a single alternative (feature-level subgoal) for each of the object-level goals based on the confidence measures, resulting in the choices for the three goals as: (H2-I1; H7-I3), (H5-J3; H6-J1) and (H5-K3; H6-K1). The constraint propagation rules are now applied. The "Double-Insert K into H" goal with the choice of mating features as (H5-K3; H6-K1) requires that features H5 and H6 be accessible after (G-2) has been accomplished. However, this condition is violated by the choice of mating-features for (G-2). So, the planner abandons this choice of feature-level subgoal for (G-2) and picks (H4-J3; H5-J1) as an alternative. This choice also is detected to conflict with the

preconditions of (G-3) so it is abandoned and another alternative is picked from the list of choice-points, namely, (H3-J3; H4-J1). Constraint propagation with this set of choices indicates no conflict so that the feature-level subgoals chosen before further refinement of the goals proceeds are (H2-I1; H7-I3), (H3-J3; H4-J1) and (H5-K3; H6-K1).

6. Future Work

Our current implementation has the first level of refinement reasonably well worked out, although it is extensible by addition of further heuristics and constraints. The spatial reasoning component needs to be developed. Specifically, the planner needs to be able to deduce any positioning constraints that arise from the need to simutaneously satify two or more spatial relationships and recognize any contradictory requirements. For this purpose, we intend to use the RAPT system [27] developed at the University of Edinburgh.

The geometric reasoning component of the system currently is able to reason about the accessibility of features based on prior operations. The feature relationships represented in the *feature graph* and models of operations are the essential ingredients to perform this reasoning. However, the treatment of geometric reasoning in the current implementation is not sufficiently general to enable the development of models for the subassembly that are at the same level of detail as the models of the objects themselves. For the present, we would like to implement the more general treatment of creation and deletion of features detailed in [17].

The specification of object models in the current implementation requires that each object in the workspace be described in detail individually. However, assembly situations often involve a number of identical objects (as is seen in Arbib's example and Torras's example of the previous section) or objects that belong in the same class. For example, bolts of different sizes still have the same set of features and toplogical relations between features although the dimension of these features may differ. Thus, we would like to structure the object database to represent a hierarchy of classes of objects with the physical objects present in the workspace as specific instances of these classes.

Based on the commands output by the strategic planner, the path planner would perform the necessary analysis to arrive at collision-free paths while the tactical planner carries out the necessary analysis to further refine and execute grasp and manipulation commands. The primitives in each of these categories are themselves not independent and therefore we need to build into the planner the ability to recognize and analyze the

inter-dependencies between the parameters selected for the various primitives comprising a task. As the commands are further refined by the tactical and path planners, interactions that were not apparent at the strategic planner level may emerge. When the interactions are sufficient to cause the plan to fail, the planner should be able to recognize the nature of the failure and replan accordingly. Development of this phase of the planner will require sensory information and we need to use appropriate models of sensors.

Last but not least, there are some necessary and easily achievable extensions to our current implementation. These are listed below.

1. The constraint propagation mechanism is uni-directional at the moment, propagating constraints always from succeeding goals to preceding goals. It should really work both ways, with the projected effects of prior actions propagated to succeeding actions as well.

2. Another important extension would be to record the decisions made by the planner such as the conflicting choices of feature level subgoals and where the conflict arose, so subsequent choices can be made in a more intelligent fashion. The *goal structure* (GOST) representation used by Tate in NONLIN[31] would be appropriate for this purpose.

Acknowledgements

We would like to thank Dr. Carme Torras for valuable discussions in formulating the approach to the problem. Michael Arbib thanks the researchers and staff at the Institute of Cybernetics of the Polytechnic University of Catalonia for their support and hospitality during the early stages of this work. Special thanks are due to Damian Lyons who actively participated in arriving at the overall system architecture and the interfaces between the strategic, tactical and path planners. We would like to thank Yanxi Liu and Thea Iberall who gave helpful suggestions based on an earlier draft of this paper.

REFERENCES

[1] Arbib, M.A., Iberall, T., and Lyons, D. (1985). Coordinated Control Programs for Movements of the Hand. *Experimental Brain Research Supplement 10*, 111-129.

[2] Begej, S. (1985). A Tactile Sensing System and Optical Tactile-Sensor Array for Robotic Applications. Technical Report 85-06, Laboratory for Perceptual Robotics, Computer and Information Science Department, University of Massachusetts, Amherst.

[3] Brooks, R.A. (1982). Symbolic Error Analysis and Robot Planning. *The International Journal of Robotics Research*, 1, 29-68.

[4] Brooks, R.A. and T. Lozano-Perez (1983). A Subdivision Algorithm In Configuration Space for Findpath with Rotation. *IJCAI 8*, Karlsruhe, Germany.

[5] Brooks, R.A. (1983). Solving the Find-Path Problem by Good Representation of Free Space, *IEEE Transactions on Systems, Man and Cybernetics*, SMC-13, 190-197.

[6] Brooks, R.A. (1983). Planning Collision Free Motions for Pick and Place Operations, *First International Symposium on Robotics Research*, August, 5-37.

[7] CAM-I (1981). CAM-I's Illustrated Glossary of Workpiece Form Features, Revised May, 1981, R-80-PPP-02.1, CAM-I, Inc., Arlington, Texas.

[8] Corkill, D.D. and V.R.Lesser (1981). A Goal-Directed Hearsay-II Architecture: Unifying Data-directed and Goal-directed Control. Technical Report 81-15, Computer and Information Science Department, University of Massachusetts, Amherst.

[9] Dufay, B. and J.C. Latombe (1983). An Approach to Automatic Robot Programming based on Inductive Learning, *First International Symposium on Robotics Research*, August, 97-115.

[10] Erman, L.D., F.Hayes-Roth, V.R.Lesser and D.R.Reddy, (1980). The HEARSAY-II Speech Understanding System: Integrating knowledge to resolve uncertainty. *Computing Surveys* 12, 213-253.

[11] Fikes, R.E. and N.J.Nilsson (1971). STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 3, 251-208.

[12] Finkel, R., R. Taylor, R. Bolles, R. Paul, J. Feldman (1975). An Overview of AL, A Programming System for Automation. Proceedings of the *Fourth International Joint Conference on Artificial Intelligence*, 1975, 758-765.

[13] Laugier, C. (1981). A Program for automatic grasping of objects with a robot arm, *Eleventh International Symposium on Industrial Robots*, Tokyo, Japan, October.

[14] Laugier, C. and J. Pertin (1983). Automatic Grasping: A Case Study in Accessibility Analysis, *International meeting on Advanced Software in Robotics*, Liege, May.

[15] Laugier, C. and J. Pertin (1985). SHARP: A System for Automatic Programming of Manipulation robots, *Third International Symposium on Robotics Research*, October.

[16] Lieberman, L.I., M.A. Wesley (1977). AUTOPASS: An Automatic Programming System for Computer Controlled Mechanical Assembly. *IBM Journal of Research and Development*, Volume 21, Number 4, 321-333.

[17] Liu, Y., (1986). A Planner for Sensor-Based Robotics in the Assembly Domain, Technical Report (in preparation), Department of Computer and Information Science, Umniversity of Massachusetts, Amherst.

[18] Lozano-Perez, T. (1976). The Design of a Mechanical Assembly System, MIT Artificial Intelligence Laboratory, TR 397, December.

[19] Lozano-Perez, T., and Winston,P.H. (1977). LAMA: A Language for Automatic Mechanical Assembly, Proceedings of the *Fifth International Joint Conference on Artificial Intelligence*, Cambridge, Massachusetts, August, 710-716.

[20] Lozano-Perez, T. (1981). Automatic Planning of Manipulator Transfer Movements, *IEEE Transactions on Systems, Man and Cybernetics*, SMC-11, 681-698.

[21] Lozano-Perez, T., Mason, M.T., and Taylor, R.H. (1984). Automatic Synthesis of Fine-Motion Strategies for Robots, *International Journal of Robotics Research*, Vol. 3, No. 1.

[22] Lozano-Perez, T. and R.A. Brooks (1985). An Approach to Automatic Robot Programming, A.I. Memo 842, MIT Artificial Intelligence Laboratory, April.

[23] Lyons, D. (1985). A Simple Set of Grasp for a Dextrous hand, Proceedings of the *1985 Robotics and Automation Conference*, March 25-28, St. Louis, Missouri.

[24] Mason, M.T. (1984). Automatic planning of Fine Motions: Correctness and Completeness, *IEEE International Conference on Robotics*, Atlanta, Georgia.

[25] Overton, K.J. (1984). The Acquisition, Processing, and Use of Tactile Sensor Data in Robot Control. Ph.D. Thesis, COINS Technical Report 84-08, University of Massachusetts, Amherst.

[26] Paul, R.P. (1981). *Robot Manipulators: Mathematics, Programming and Control*, The MIT Press, Cambridge, Massachusetts.

[27] Popplestone, R.J., A.P.Ambler and I.Bellos (1980). An Interpreter for a Language for Describing Assemblies, *Artificial Intelligence*, 14, 79-107.

[28] Sacerdoti, E.D. (1977). *A Structure for Plans and Behavior*, New York:Elsevier North-Holland.

[29] Stefik, M. (1981). Planning with Constraints (MOLGEN: Part 1). *Artificial Intelligence*, 6, 111-140.

[30] Stefik, M. (1981). Planning and Meta-Planning (MOLGEN: Part 2). *Artificial Intelligence*, 6, 141-170.

[31] Tate, A. (1977). Generating Project Networks. Proceedings of the *Fifth International Joint Conference on Artificial Intelligence*, Cambridge, Mass., August, 888-893.

[32] Taylor, R.H. (1976). The Synthesis of Manipulator Control Programs from Task-level Specifications, Stanford Artificial Intelligence Laboratory, AIM-282, July.

[33] Torras, C. and F. Thomas (1985). Planning with Constraints: Application to Sensor-Based Robot Assembly Tasks. Proceedings of the IFAC *Symposium on Robot Control*, November 6-8, Barcelona, Spain, 453-456.

[34] Udupa, S.M. (1977). Collision Detection and Avoidance in Computer Controlled Manipulators, Proceedings of the *Fifth International Joint Conference on Artificial Intelligence*, Cambridge, Massachusetts, August, 737-748.

[35] Vijaykumar, R., Michael A. Arbib, and Yanxi Liu. (1985). Dynamic Planning for Sensor-Based Robots. Proceedings of the IFAC *Symposium on Robot Control*, November 6-8, Barcelona, Spain, 401-406.

[36] Wilkins, D. (1983). Representation in a Domain-Independent Planner. *IJCAI 8*, 2, 733-740.

APPENDIX A

OBJECT REPRESENTATIONS FOR EXAMPLE 1: HUBER'S EXAMPLE

```
;;; -*- Mode: LISP -*-
;;; This file contains the definitions of the four parts that are in Huber's first
;;; example and the significant features of these parts.


;;;*****************************************************************************
;;;                    HUBER'S FIRST EXAMPLE
;;;*****************************************************************************
;;; Each of the parts and its features are grouped together and described
;;; below:
;;;
;;;                    ---------------
;;;                    |   PART A    |
;;;                    ---------------

(setf (get 'part-a 'frame) (make-object
                             object-shape 'cylinder
                             object-length 3.6
                             object-width  3.6
                             object-height 2.4
                             object-features  (list  'a1 'a2 'a3 'a4)))

;;; smaller end of A

(setf (get 'a1 'frame) (make-cylindrical-hole
                         adjacent-features (list 'a2)
                         internal-diameter 1.0
                         depth 2.4
                         cy-h-normal-axis 'neg-h-axis))

;;; threads at a1

(setf (get 'a2 'frame)  (make-thread
                          adjacent-features (list 'a1)
                          thr-diameter 1.0
                          internal-thr 'true
                          thread-width 0.5
                          thread-pitch 0.1))

;;; larger end of A

(setf (get 'a3 'frame) (make-cylindrical-hole
                         adjacent-features (list 'a4)
                         internal-diameter 3.6
                         depth 2.0
                         cy-h-normal-axis 'h-axis))

;;; threads at a3

(setf (get 'a4 'frame)  (make-thread
                          adjacent-features (list 'a3)
                          thr-diameter 3.6
                          internal-thr 'false
                          thread-width 0.5
                          thread-pitch 0.25))


;;;Part B is itself an assembly of two subparts, one shaped like a cuboid and
;;;another shaped like a cylinder. Thus, one end of B (B1) is flat and rectangular
;;;and the opposite end (B2) is circular and threaded. The part is symmetric
;;;about the axis normal to sides B1 and B2. There are a pair of grooves on the
;;;smaller sides of B1 across from each other.

;;;                    ---------------
;;;                    |    PART B   |
;;;                    ---------------


(setf (get 'part-b 'frame) (make-object
                             object-shape 'box-plus-cylinder
                             object-length 3.4
                             object-width  3.1
                             object-height 2.6
                             object-features  (list 'b1 'b2 'b3 'b4)))
;;; box end of B

(setf (get 'b1 'frame) (make-rectangular-face
                         face-length 3.4
                         face-width 1.6
                         p-f-normal-axis 'w-axis))

;;; other end of B
```

```
(setf (get 'b2 'frame) (make-cylindrical-hole
                         adjacent-features (list 'b3)
                         internal-diameter 2.9
                         depth 2.7
                         cy-h-normal-axis 'neg-w-axis))

;;; threads at the end b2

(setf (get 'b3 'frame) (make-thread
                         adjacent-features (list 'b2 'b4)
                         thr-diameter 2.9
                         internal-thr 'false
                         thread-width 1.3
                         thread-pitch 0.4))

;;; side of B

(setf (get 'b4 'frame) (make-cylindrical-surface
                         adjacent-features (list 'b3)
                         cy-height 3.1
                         cy-diameter 2.6
                         cy-s-normal-axis 'h-axis))


;;;Part C looks like a cylinder that is open at both ends except that it is wider
;;;at one end than at the other. Its two ends, C1 and C2, are both circular.
;;;The side C1 is
;;;larger in diameter than side C2. C1 is threaded while C2 is not. The part is
;;;symmetric about the axis normal to the sides C1 and C2.
;;;
;;;                          ------------------
;;;                          |     PART C     |
;;;                          ------------------

(setf (get 'part-c 'frame) (make-object
                             object-shape 'cylinder
                             object-length 3.6
                             object-width  3.6
                             object-height 2.5
                             object-features (list 'c1 'c2 'c3)))

;;; wider end of C

(setf (get 'c1 'frame) (make-cylindrical-hole
                         adjacent-features (list 'c2)
                         internal-diameter 3.6
                         depth 2.5
                         cy-h-normal-axis 'h-axis))

;;; threads at c1

(setf (get 'c2 'frame) (make-thread
                         adjacent-features (list 'c1)
                         thr-diameter 3.6
                         internal-thr 'true
                         thread-width 0.5
                         thread-pitch 0.25))

;;; narrower end of C

(setf (get 'c3 'frame) (make-cylindrical-hole
                         internal-diameter 3.3
                         depth 2.5
                         cy-h-normal-axis 'neg-h-axis))


;;;Part D looks like a hollow cylinder with a small height and the two ends of
;;;the cylinder are named  D1 and D2. The part is threaded internally all the way
;;;from side D1 to D2.
;;;This part is also symmetric about the
;;;axis normal to the sides D1 and D2. There is an obtruding lip from D1
;;;which distinguishes the two ends D1 and D2.
;;;
;;;                          ------------------
;;;                          |     PART D     |
;;;                          ------------------

(setf (get 'part-d 'frame) (make-object
                             object-shape 'cylinder
                             object-length 3.5
                             object-width  3.5
                             object-height 1.4
```

```
                        object-features (list 'd1 'd2 'd3 'd4)))

;;; lip of D

(setf (get 'd1 'frame) (make-cylindrical-hole
                         adjacent-features (list 'd2)
                         overlapping-features (list 'd3)
                         internal-diameter 2.9
                         depth 1.4
                         cy-h-normal-axis 'h-axis))

;;; threads at d1

(setf (get 'd2 'frame) (make-thread
                         adjacent-features (list 'd1)
                         overlapping-features (list 'd4)
                         thr-diameter 2.9
                         internal-thr 'true
                         thread-width 1.3
                         thread-pitch 0.4))

;;; other end of D

(setf (get 'd3 'frame) (make-cylindrical-hole
                         adjacent-features (list 'd4)
                         overlapping-features (list 'd1)
                         internal-diameter 2.9
                         depth 1.4
                         cy-h-normal-axis 'neg-h-axis))

;;; thread at d3

(setf (get 'd4 'frame) (make-thread
                         adjacent-features (list 'd3)
                         overlapping-features (list 'd2)
                         thr-diameter 2.9
                         internal-thr 'true
                         thread-width 1.3
                         thread-pitch 0.4))
```

APPENDIX B

OBJECT REPRESENTATIONS FOR EXAMPLE 2: ARBIB'S EXAMPLE

```
;;;*****************************************************************
;;;                      ARBIB'S EXAMPLE
;;;*****************************************************************

(setf (get 'part-e 'frame) (make-object
                             object-shape 'circular-plate
                             object-length 5.0
                             object-width  5.0
                             object-height 1.0
                             object-features  (list  'e1 'e2 'e3)))


(setf (get 'e1 'frame) (make-circular-face
                         inclusive-features (list 'e2 'e3)
                         p-f-diameter 5.0
                         normal-axis 'h-axis))

(setf (get 'e2 'frame)  (make-cylindrical-hole
                          encompassing-feature  'e1
                          internal-diameter 0.5
                          depth 0.8
                          cy-h-normal-axis 'h-axis))


(setf (get 'e3 'frame) (make-cylindrical-hole
                         encompassing-feature  'e1
                         internal-diameter 0.5
                         depth 0.8
                         cy-h-normal-axis 'h-axis))

;;;*****************************************************


(setf (get 'part-f 'frame) (make-object
                             object-shape 'cylinder
                             object-length 4.0
                             object-width 0.5
                             object-height 0.5
                             object-features (list 'f1 'f2 'f3)))

(setf (get 'f1 'frame)  (make-cylindrical-surface
                          adjacent-features (list 'f2 'f3)
                          cy-height 4.0
                          cy-diameter 0.5
                          cy-s-normal-axis 'l-axis))


(setf (get 'f2 'frame)  (make-circular-face
                          adjacent-features (list 'f1)
                          p-f-diameter 0.5
                          normal-axis 'l-axis))


(setf (get 'f3 'frame)  (make-circular-face
                          adjacent-features (list 'f1)
                          p-f-diameter 0.5
                          normal-axis 'neg-l-axis))


;;;*****************************************************


(setf (get 'part-g 'frame) (make-object
                             object-shape 'cylinder
                             object-length 4.0
                             object-width 0.5
                             object-height 0.5
                             object-features (list 'g1 'g2 'g3)))

(setf (get 'g1 'frame)  (make-cylindrical-surface
                          adjacent-features (list 'g2 'g3)
                          cy-height 4.0
                          cy-diameter 0.5
                          cy-s-normal-axis 'l-axis))


(setf (get 'g2 'frame)  (make-circular-face
                          adjacent-features (list 'g1)
                          p-f-diameter 0.5
```

```
                    normal-axis 'l-axis))

(setf (get 'g3 'frame)  (make-circular-face
                          adjacent-features (list 'g1)
                          p-f-diameter 0.5
                          normal-axis 'neg-l-axis))
```

APPENDIX C

OBJECT REPRESENTATIONS FOR EXAMPLE 3: TORRAS'S EXAMPLE

```
;;; -*- Mode: LISP -*-

;;;*****************************************************************************
;;;                        CARMEN'S EXAMPLE
;;;*****************************************************************************


(setf (get 'part-h 'frame)  (make-object
                              object-length 6.0
                              object-width 6.0
                              object-height 1.0
                              object-features (list 'h1 'h2 'h3
                                                    'h4 'h5 'h6
                                                    'h7)))

(setf (get 'h1 'frame)  (make-rectangular-face
                          inclusive-features (list 'h2 'h3 'h4
                                                   'h5 'h6 'h7)
                          face-length 6.0
                          face-width 6.0))

(setf (get 'h2 'frame) (make-cylindrical-hole
                         encompassing-feature 'h1
                         internal-diameter  2.0
                         depth 1.0))

(setf (get 'h3 'frame) (make-cylindrical-hole
                         encompassing-feature 'h1
                         internal-diameter  0.5
                         depth 1.0))

(setf (get 'h4 'frame) (make-cylindrical-hole
                         encompassing-feature 'h1
                         internal-diameter  0.5
                         depth 1.0))

(setf (get 'h5 'frame) (make-cylindrical-hole
                         encompassing-feature 'h1
                         internal-diameter  0.5
                         depth 1.0))

(setf (get 'h6 'frame) (make-cylindrical-hole
                         encompassing-feature 'h1
                         internal-diameter  0.5
                         depth 1.0))

(setf (get 'h7 'frame) (make-cylindrical-hole
                         encompassing-feature 'h1
                         internal-diameter  0.5
                         depth 1.0))


;;;*****************************************************************

(setf (get 'part-i 'frame)  (make-object
                              object-length 3.0
                              object-width 2.0
                              object-height 2.0
                              object-features (list 'i1 'i2 'i3)))

(setf (get 'i1 'frame)  (make-cylindrical-shaft
                          adjacent-features (list 'i2)
                          diameter 2.0
                          height 2.0))

(setf (get 'i2 'frame)  (make-cylindrical-surface
                          adjacent-features (list 'i1 'i3)
                          cy-height 0.5
                          cy-diameter 0.5))

(setf (get 'i3 'frame) (make-cylindrical-shaft
                          adjacent-features (list 'i2)
                          diameter 0.5
                          height 2.0))

;;;*****************************************************************


(setf (get 'part-j 'frame)  (make-object
                              object-length 2.0
                              object-width 1.5
```

```
                        object-height 0.5
                        object-features (list 'j1 'j2 'j3)))

(setf (get 'j1 'frame) (make-cylindrical-shaft
                         adjacent-features (list 'j2)
                         diameter 0.5
                         height 2.0))

(setf (get 'j2 'frame)  (make-cylindrical-surface
                         adjacent-features (list 'j1 'j3)
                         cy-height 0.5
                         cy-diameter 0.5))

(setf (get 'j3 'frame) (make-cylindrical-shaft
                         adjacent-features (list 'j2)
                         diameter 0.5
                         height 2.0))

;;;**************************************************************

(setf (get 'part-k 'frame)  (make-object
                              object-length 2.0
                              object-width 1.5
                              object-height 0.5
                              object-features (list 'k1 'k2 'k3)))

(setf (get 'k1 'frame) (make-cylindrical-shaft
                         adjacent-features (list 'k2)
                         diameter 0.5
                         height 2.0))


(setf (get 'k2 'frame)  (make-cylindrical-surface
                         adjacent-features (list 'k1 'k3)
                         cy-height 0.5
                         cy-diameter 0.5))


(setf (get 'k3 'frame) (make-cylindrical-shaft
                         adjacent-features (list 'k2)
                         diameter 0.5
                         height 2.0))
```

APPENDIX D

PROGRAM TRACES

```
(load '[robot.planner]test.lisp)
Loading GRAFIX:RBT$DISK:[ROBOT.PLANNER]TEST.LISP into package USER
Loading GRAFIX:RBT$DISK:[ROBOT.PLANNER]HFEXAMPLE.LISP into package USER
Loading GRAFIX:RBT$DISK:[ROBOT.PLANNER]MAIN.LISP into package USER


--------------------------------------------------
                  Program Input
--------------------------------------------------

List of Goals

1 : (FIT PART-B IN PART-A)

2 : (THREAD PART-C WITH PART-A)

3 : (THREAD PART-D WITH PART-B)


Ordering Constraints

Goal 1 precedes Goal 2.
Goal 2 precedes Goal 3.


--------------------------------------------------
Initial selection of feature level subgoals completed
--------------------------------------------------


After applying the constraints associated with the operations
the feature level subgoals for each object level goal along with its
confidence rate are:


Object Level Goal:   (FIT PART-B IN PART-A).

      Feature Level Subgoals                                  Confidence Rates

      (FIT FEATURE B1 OF PART-B IN FEATURE A3 OF PART-A)              0.33333334

      (FIT FEATURE B2 OF PART-B IN FEATURE A3 OF PART-A)              0.33333334

      (FIT FEATURE B4 OF PART-B IN FEATURE A3 OF PART-A)              0.33333334

Object Level Goal:   (THREAD PART-C WITH PART-A).

      Feature Level Subgoals                                  Confidence Rates

      (THREAD FEATURE C2 OF PART-C WITH FEATURE A4 OF PART-A)               1.0

Object Level Goal:   (THREAD PART-D WITH PART-B).

      Feature Level Subgoals                                  Confidence Rates

      (THREAD FEATURE D2 OF PART-D WITH FEATURE B3 OF PART-B)               0.5

      (THREAD FEATURE D4 OF PART-D WITH FEATURE B3 OF PART-B)               0.5


--------------------------------------------------
          Application of Heuristic Rules
--------------------------------------------------

Heuristic closeness-of-fit applies to object level goal
 (FIT PART-B IN PART-A) and modifies the confidence
 rates of subgoals as follows:

      (FIT FEATURE B1 OF PART-B IN FEATURE A3 OF PART-A)              0.6666667

      (FIT FEATURE B2 OF PART-B IN FEATURE A3 OF PART-A)              0.16666667

      (FIT FEATURE B4 OF PART-B IN FEATURE A3 OF PART-A)              0.16666667

Heuristic find-distinguishing-feature  applies to feature level subgoal:

            (FIT FEATURE B1 OF PART-B IN FEATURE A3 OF PART-A).

 The confidence rates of all the feature-level subgoals of
(FIT PART-B IN PART-A) are modified as follows:

      (FIT FEATURE B1 OF PART-B IN FEATURE A3 OF PART-A)              0.8333334
```

(FIT FEATURE B2 OF PART-B IN FEATURE A3 OF PART-A) 0.083333336

(FIT FEATURE B4 OF PART-B IN FEATURE A3 OF PART-A) 0.083333336

Subgoal Selection

Based on the confidence rates associated with feature level subgoals
(FIT FEATURE B1 OF PART-B IN FEATURE A3 OF PART-A) has been chosen as
the most promising option for goal (FIT PART-B IN PART-A)

Based on the confidence rates associated with feature level subgoals
(THREAD FEATURE D4 OF PART-D WITH FEATURE B3 OF PART-B) has been chosen as
the most promising option for goal (THREAD PART-D WITH PART-B)

Constraint Analysis

The accessibility constraints that result from the current
choice of mating features for each of the object level goals are:

Object Level Goal: (FIT PART-B IN PART-A).

Feature Level Subgoal: (FIT FEATURE B1 OF PART-B IN FEATURE A3 OF PART-A).

Accessibility preconditions: (ACCESSIBLE A3) (ACCESSIBLE B1)

Object Level Goal: (THREAD PART-C WITH PART-A).

Feature Level Subgoal: (THREAD FEATURE C2 OF PART-C WITH FEATURE A4 OF PART-A).

Accessibility preconditions: (ACCESSIBLE C2) (ACCESSIBLE A4)

Object Level Goal: (THREAD PART-D WITH PART-B).

Feature Level Subgoal: (THREAD FEATURE D4 OF PART-D WITH FEATURE B3 OF PART-B).

Accessibility preconditions: (ACCESSIBLE D4) (ACCESSIBLE B3)

The accessibility condition (ACCESSIBLE A4) is propagated forward
from goal (THREAD FEATURE C2 OF PART-C WITH FEATURE A4 OF PART-A) and gets added to the
set of postconditions of goal (FIT PART-B IN PART-A).

The planner checks to see if the execution of
(FIT FEATURE B1 OF PART-B IN FEATURE A3 OF PART-A) will violate
any of the postconditions associated with (FIT PART-B IN PART-A).

Request the user for information:

The goal is to execute the operation (FIT FEATURE B1 OF PART-B IN FEATURE A3 OF PART-A).
Indicate if the feature A4 will be accessible after execution of this operation
by typing yes or no:

yes

Tentative Plan at the Feature Level

Object Level Goal: (FIT PART-B IN PART-A).

Feature Level Subgoals

(FIT FEATURE B1 OF PART-B IN FEATURE A3 OF PART-A).

Object Level Goal: (THREAD PART-C WITH PART-A).

Feature Level Subgoals

(THREAD FEATURE C2 OF PART-C WITH FEATURE A4 OF PART-A).

Object Level Goal: (THREAD PART-D WITH PART-B).

Feature Level Subgoals

```
(THREAD FEATURE D4 OF PART-D WITH FEATURE B3 OF PART-B).
#<VMS4-PATHNAME "GRAFIX:RBT$DISK:[ROBOT.PLANNER]TEST.LISP">
NIL
(dribble-end)
```

```
(load '[robot.planner]test1.lisp)
Loading GRAFIX:RBT$DISK:[ROBOT.PLANNER]TEST1.LISP into package USER
Loading GRAFIX:RBT$DISK:[ROBOT.PLANNER]MAIN.LISP into package USER
Loading GRAFIX:RBT$DISK:[ROBOT.PLANNER]AFE.LISP into package USER


--------------------------------------------------
                  Program Input
--------------------------------------------------

List of Goals

1 : (FIT PART-F IN PART-E)

2 : (FIT PART-G WITH PART-E)

Ordering Constraints

Goal 1 precedes Goal 2.

--------------------------------------------------
Initial selection of feature level subgoals completed
--------------------------------------------------

After applying the constraints associated with the operations
the feature level subgoals for each object level goal along with its
confidence rate are:

Object Level Goal:    (FIT PART-F IN PART-E).

      Feature Level Subgoals                                  Confidence Rates

      (FIT FEATURE F2 OF PART-F IN FEATURE E2 OF PART-E)          0.25

      (FIT FEATURE F2 OF PART-F IN FEATURE E3 OF PART-E)          0.25

      (FIT FEATURE F3 OF PART-F IN FEATURE E2 OF PART-E)          0.25

      (FIT FEATURE F3 OF PART-F IN FEATURE E3 OF PART-E)          0.25

Object Level Goal:    (FIT PART-G WITH PART-E).

      Feature Level Subgoals                                  Confidence Rates

      (FIT FEATURE G2 OF PART-G WITH FEATURE E2 OF PART-E)        0.25

      (FIT FEATURE G2 OF PART-G WITH FEATURE E3 OF PART-E)        0.25

      (FIT FEATURE G3 OF PART-G WITH FEATURE E2 OF PART-E)        0.25

      (FIT FEATURE G3 OF PART-G WITH FEATURE E3 OF PART-E)        0.25

--------------------------------------------------
            Application of Heuristic Rules
--------------------------------------------------

Feature level subgoal (FIT FEATURE G3 OF PART-G WITH FEATURE E2 OF PART-E)
is being moved to the list of choice points associated with goal
(FIT PART-G WITH PART-E) as there is a symmetrical alternative in
(FIT FEATURE G3 OF PART-G WITH FEATURE E3 OF PART-E).

Feature level subgoal (FIT FEATURE G2 OF PART-G WITH FEATURE E3 OF PART-E)
is being moved to the list of choice points associated with goal
(FIT PART-G WITH PART-E) as there is a symmetrical alternative in
(FIT FEATURE G3 OF PART-G WITH FEATURE E3 OF PART-E).

Feature level subgoal (FIT FEATURE G2 OF PART-G WITH FEATURE E2 OF PART-E)
is being moved to the list of choice points associated with goal
(FIT PART-G WITH PART-E) as there is a symmetrical alternative in
(FIT FEATURE G3 OF PART-G WITH FEATURE E3 OF PART-E).

Feature level subgoal (FIT FEATURE F3 OF PART-F IN FEATURE E2 OF PART-E)
is being moved to the list of choice points associated with goal
(FIT PART-F IN PART-E) as there is a symmetrical alternative in
(FIT FEATURE F3 OF PART-F IN FEATURE E3 OF PART-E).

Feature level subgoal (FIT FEATURE F2 OF PART-F IN FEATURE E3 OF PART-E)
```

```
is being moved to the list of choice points associated with goal
(FIT PART-F IN PART-E) as there is a symmetrical alternative in
(FIT FEATURE F3 OF PART-F IN FEATURE E3 OF PART-E).

Feature level subgoal (FIT FEATURE F2 OF PART-F IN FEATURE E2 OF PART-E)
is being moved to the list of choice points associated with goal
(FIT PART-F IN PART-E) as there is a symmetrical alternative in
(FIT FEATURE F3 OF PART-F IN FEATURE E3 OF PART-E).


--------------------------------------------------
                Constraint Analysis
--------------------------------------------------


The accessibility constraints that result from the current
choice of mating features for each of the object level goals are:

Object Level Goal:   (FIT PART-F IN PART-E).

Feature Level Subgoal:   (FIT FEATURE F3 OF PART-F IN FEATURE E3 OF PART-E).

Accessibility preconditions:     (ACCESSIBLE E3)    (ACCESSIBLE F3)

Object Level Goal:   (FIT PART-G WITH PART-E).

Feature Level Subgoal:   (FIT FEATURE G3 OF PART-G WITH FEATURE E3 OF PART-E).

Accessibility preconditions:     (ACCESSIBLE E3)    (ACCESSIBLE G3)

The accessibility condition (ACCESSIBLE E3) is propagated forward
from goal (FIT FEATURE G3 OF PART-G WITH FEATURE E3 OF PART-E) and gets added to the
set of postconditions of goal (FIT PART-F IN PART-E).

The planner checks to see if the execution of
(FIT FEATURE F3 OF PART-F IN FEATURE E3 OF PART-E) will violate
any of the postconditions associated with (FIT PART-F IN PART-E).

Reasoning about the geometry of the features indicates that feature E3
will be eliminated by executing the subgoal (FIT FEATURE F3 OF PART-F IN FEATURE E3 OF PART-E).

An alternative feature subgoal (FIT FEATURE F3 OF PART-F IN FEATURE E2 OF PART-E)
is chosen for the object level goal (FIT PART-F IN PART-E) to avoid
the earlier conflict that arose from the constraint propagation.


--------------------------------------------------
         Tentative Plan at the Feature Level
--------------------------------------------------


Object Level Goal:   (FIT PART-F IN PART-E).

      Feature Level Subgoals

      (FIT FEATURE F3 OF PART-F IN FEATURE E2 OF PART-E).

Object Level Goal:   (FIT PART-G WITH PART-E).

      Feature Level Subgoals

      (FIT FEATURE G3 OF PART-G WITH FEATURE E3 OF PART-E).
#<VMS4-PATHNAME "GRAFIX:RBT$DISK:[ROBOT.PLANNER]TEST1.LISP">
NIL
(dribble-end)
```

```
(load '[robot.planner]test2.lisp)
Loading GRAFIX:RBT$DISK:[ROBOT.PLANNER]TEST2.LISP into package USER
Loading GRAFIX:RBT$DISK:[ROBOT.PLANNER]CARME.LISP into package USER
Loading GRAFIX:RBT$DISK:[ROBOT.PLANNER]MAIN.LISP into package USER
```

Program Input

List of Goals

1 : (DOUBLE-INSERT PART-I IN PART-H)

2 : (DOUBLE-INSERT PART-J IN PART-H)

3 : (DOUBLE-INSERT PART-K IN PART-H)

Ordering Constraints

Goal 1 precedes Goal 2.
Goal 2 precedes Goal 3.

Initial selection of feature level subgoals completed

After applying the constraints associated with the operations
the feature level subgoals for each object level goal along with its
confidence rate are:

Object Level Goal: (DOUBLE-INSERT PART-I IN PART-H).

Feature Level Subgoals	Confidence Rates
(DOUBLE-INSERT FEATURE I1 OF PART-I IN FEATURE H2 OF PART-H AND FEATURE I3 IN FEATURE H3)	0.2
(DOUBLE-INSERT FEATURE I1 OF PART-I IN FEATURE H2 OF PART-H AND FEATURE I3 IN FEATURE H4)	0.2
(DOUBLE-INSERT FEATURE I1 OF PART-I IN FEATURE H2 OF PART-H AND FEATURE I3 IN FEATURE H5)	0.2
(DOUBLE-INSERT FEATURE I1 OF PART-I IN FEATURE H2 OF PART-H AND FEATURE I3 IN FEATURE H6)	0.2
(DOUBLE-INSERT FEATURE I1 OF PART-I IN FEATURE H2 OF PART-H AND FEATURE I3 IN FEATURE H7)	0.2

Object Level Goal: (DOUBLE-INSERT PART-J IN PART-H).

Feature Level Subgoals	Confidence Rates
(DOUBLE-INSERT FEATURE J1 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J3 IN FEATURE H3)	0.0625
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J1 IN FEATURE H3)	0.0625
(DOUBLE-INSERT FEATURE J1 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J3 IN FEATURE H4)	0.0625
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J1 IN FEATURE H4)	0.0625
(DOUBLE-INSERT FEATURE J1 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J3 IN FEATURE H5)	0.0625
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J1 IN FEATURE H5)	0.0625
(DOUBLE-INSERT FEATURE J1 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J3 IN FEATURE H6)	0.0625
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J1 IN FEATURE H6)	0.0625

(DOUBLE-INSERT FEATURE J1 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J3 IN FEATURE H7)	0.0625
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J1 IN FEATURE H7)	0.0625
(DOUBLE-INSERT FEATURE J1 OF PART-J IN FEATURE H3 OF PART-H AND FEATURE J3 IN FEATURE H4)	0.0625
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H3 OF PART-H AND FEATURE J1 IN FEATURE H4)	0.0625
(DOUBLE-INSERT FEATURE J1 OF PART-J IN FEATURE H4 OF PART-H AND FEATURE J3 IN FEATURE H5)	0.0625
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H4 OF PART-H AND FEATURE J1 IN FEATURE H5)	0.0625
(DOUBLE-INSERT FEATURE J1 OF PART-J IN FEATURE H5 OF PART-H AND FEATURE J3 IN FEATURE H6)	0.0625
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H5 OF PART-H AND FEATURE J1 IN FEATURE H6)	0.0625

Object Level Goal: (DOUBLE-INSERT PART-K IN PART-H).

Feature Level Subgoals	Confidence Rates
(DOUBLE-INSERT FEATURE K1 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K3 IN FEATURE H3)	0.0625
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K1 IN FEATURE H3)	0.0625
(DOUBLE-INSERT FEATURE K1 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K3 IN FEATURE H4)	0.0625
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K1 IN FEATURE H4)	0.0625
(DOUBLE-INSERT FEATURE K1 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K3 IN FEATURE H5)	0.0625
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K1 IN FEATURE H5)	0.0625
(DOUBLE-INSERT FEATURE K1 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K3 IN FEATURE H6)	0.0625
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K1 IN FEATURE H6)	0.0625
(DOUBLE-INSERT FEATURE K1 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K3 IN FEATURE H7)	0.0625
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K1 IN FEATURE H7)	0.0625
(DOUBLE-INSERT FEATURE K1 OF PART-K IN FEATURE H3 OF PART-H AND FEATURE K3 IN FEATURE H4)	0.0625
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H3 OF PART-H AND FEATURE K1 IN FEATURE H4)	0.0625
(DOUBLE-INSERT FEATURE K1 OF PART-K IN FEATURE H4 OF PART-H AND FEATURE K3 IN FEATURE H5)	0.0625
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H4 OF PART-H AND FEATURE K1 IN FEATURE H5)	0.0625
(DOUBLE-INSERT FEATURE K1 OF PART-K IN FEATURE H5 OF PART-H AND FEATURE K3 IN FEATURE H6)	0.0625
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H5 OF PART-H AND FEATURE K1 IN FEATURE H6)	0.0625

Application of Heuristic Rules

Heuristic closeness-of-fit applies to object level goal (DOUBLE-INSERT PART-K IN PART-H) and modifies the confidence

rates of subgoals as follows:

(DOUBLE-INSERT FEATURE K1 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K3 IN FEATURE H3)	0.03125
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K1 IN FEATURE H3)	0.03125
(DOUBLE-INSERT FEATURE K1 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K3 IN FEATURE H4)	0.03125
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K1 IN FEATURE H4)	0.03125
(DOUBLE-INSERT FEATURE K1 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K3 IN FEATURE H5)	0.03125
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K1 IN FEATURE H5)	0.03125
(DOUBLE-INSERT FEATURE K1 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K3 IN FEATURE H6)	0.03125
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K1 IN FEATURE H6)	0.03125
(DOUBLE-INSERT FEATURE K1 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K3 IN FEATURE H7)	0.03125
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K1 IN FEATURE H7)	0.03125
(DOUBLE-INSERT FEATURE K1 OF PART-K IN FEATURE H3 OF PART-H AND FEATURE K3 IN FEATURE H4)	0.11458333
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H3 OF PART-H AND FEATURE K1 IN FEATURE H4)	0.11458333
(DOUBLE-INSERT FEATURE K1 OF PART-K IN FEATURE H4 OF PART-H AND FEATURE K3 IN FEATURE H5)	0.11458333
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H4 OF PART-H AND FEATURE K1 IN FEATURE H5)	0.11458333
(DOUBLE-INSERT FEATURE K1 OF PART-K IN FEATURE H5 OF PART-H AND FEATURE K3 IN FEATURE H6)	0.11458333
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H5 OF PART-H AND FEATURE K1 IN FEATURE H6)	0.11458333

Heuristic closeness-of-fit applies to object level goal (DOUBLE-INSERT PART-J IN PART-H) and modifies the confidence rates of subgoals as follows:

(DOUBLE-INSERT FEATURE J1 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J3 IN FEATURE H3)	0.03125
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J1 IN FEATURE H3)	0.03125
(DOUBLE-INSERT FEATURE J1 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J3 IN FEATURE H4)	0.03125
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J1 IN FEATURE H4)	0.03125
(DOUBLE-INSERT FEATURE J1 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J3 IN FEATURE H5)	0.03125
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J1 IN FEATURE H5)	0.03125
(DOUBLE-INSERT FEATURE J1 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J3 IN FEATURE H6)	0.03125
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J1 IN FEATURE H6)	0.03125
(DOUBLE-INSERT FEATURE J1 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J3 IN FEATURE H7)	0.03125
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J1 IN FEATURE H7)	0.03125

(DOUBLE-INSERT FEATURE J1 OF PART-J IN FEATURE H3
OF PART-H AND FEATURE J3 IN FEATURE H4) 0.11458333

(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H3
OF PART-H AND FEATURE J1 IN FEATURE H4) 0.11458333

(DOUBLE-INSERT FEATURE J1 OF PART-J IN FEATURE H4
OF PART-H AND FEATURE J3 IN FEATURE H5) 0.11458333

(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H4
OF PART-H AND FEATURE J1 IN FEATURE H5) 0.11458333

(DOUBLE-INSERT FEATURE J1 OF PART-J IN FEATURE H5
OF PART-H AND FEATURE J3 IN FEATURE H6) 0.11458333

(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H5
OF PART-H AND FEATURE J1 IN FEATURE H6) 0.11458333

Feature level subgoal (DOUBLE-INSERT FEATURE K1 OF PART-K IN FEATURE H5 OF PART-H AND FEATURE K3 IN FEATURE H6)
is being moved to the list of choice points associated with goal
(DOUBLE-INSERT PART-K IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H5 OF PART-H AND FEATURE K1 IN FEATURE H6).

Feature level subgoal (DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H4 OF PART-H AND FEATURE K1 IN FEATURE H5)
is being moved to the list of choice points associated with goal
(DOUBLE-INSERT PART-K IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H5 OF PART-H AND FEATURE K1 IN FEATURE H6).

Feature level subgoal (DOUBLE-INSERT FEATURE K1 OF PART-K IN FEATURE H4 OF PART-H AND FEATURE K3 IN FEATURE H5)
is being moved to the list of choice points associated with goal
(DOUBLE-INSERT PART-K IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H5 OF PART-H AND FEATURE K1 IN FEATURE H6).

Feature level subgoal (DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H3 OF PART-H AND FEATURE K1 IN FEATURE H4)
is being moved to the list of choice points associated with goal
(DOUBLE-INSERT PART-K IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H5 OF PART-H AND FEATURE K1 IN FEATURE H6).

Feature level subgoal (DOUBLE-INSERT FEATURE K1 OF PART-K IN FEATURE H3 OF PART-H AND FEATURE K3 IN FEATURE H4)
is being moved to the list of choice points associated with goal
(DOUBLE-INSERT PART-K IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H5 OF PART-H AND FEATURE K1 IN FEATURE H6).

Feature level subgoal (DOUBLE-INSERT FEATURE K1 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K3 IN FEATURE H7)
is being moved to the list of choice points associated with goal
(DOUBLE-INSERT PART-K IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K1 IN FEATURE H7).

Feature level subgoal (DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K1 IN FEATURE H6)
is being moved to the list of choice points associated with goal
(DOUBLE-INSERT PART-K IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K1 IN FEATURE H7).

Feature level subgoal (DOUBLE-INSERT FEATURE K1 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K3 IN FEATURE H6)
is being moved to the list of choice points associated with goal
(DOUBLE-INSERT PART-K IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K1 IN FEATURE H7).

Feature level subgoal (DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K1 IN FEATURE H5)
is being moved to the list of choice points associated with goal
(DOUBLE-INSERT PART-K IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K1 IN FEATURE H7).

Feature level subgoal (DOUBLE-INSERT FEATURE K1 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K3 IN FEATURE H5)
is being moved to the list of choice points associated with goal
(DOUBLE-INSERT PART-K IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K1 IN FEATURE H7).

Feature level subgoal (DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K1 IN FEATURE H4)
is being moved to the list of choice points associated with goal
(DOUBLE-INSERT PART-K IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K1 IN FEATURE H7).

Feature level subgoal (DOUBLE-INSERT FEATURE K1 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K3 IN FEATURE H4)
is being moved to the list of choice points associated with goal
(DOUBLE-INSERT PART-K IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K1 IN FEATURE H7).

Feature level subgoal (DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K1 IN FEATURE H3)
is being moved to the list of choice points associated with goal
(DOUBLE-INSERT PART-K IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K1 IN FEATURE H7).

Feature level subgoal (DOUBLE-INSERT FEATURE K1 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K3 IN FEATURE H3)
is being moved to the list of choice points associated with goal
(DOUBLE-INSERT PART-K IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H2 OF PART-H AND FEATURE K1 IN FEATURE H7).

Feature level subgoal (DOUBLE-INSERT FEATURE J1 OF PART-J IN FEATURE H5 OF PART-H AND FEATURE J3 IN FEATURE H6)
is being moved to the list of choice points associated with goal
(DOUBLE-INSERT PART-J IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H5 OF PART-H AND FEATURE J1 IN FEATURE H6).

Feature level subgoal (DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H4 OF PART-H AND FEATURE J1 IN FEATURE H5)
is being moved to the list of choice points associated with goal
(DOUBLE-INSERT PART-J IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H5 OF PART-H AND FEATURE J1 IN FEATURE H6).

Feature level subgoal (DOUBLE-INSERT FEATURE J1 OF PART-J IN FEATURE H4 OF PART-H AND FEATURE J3 IN FEATURE H5)
is being moved to the list of choice points associated with goal
(DOUBLE-INSERT PART-J IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H5 OF PART-H AND FEATURE J1 IN FEATURE H6).

Feature level subgoal (DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H3 OF PART-H AND FEATURE J1 IN FEATURE H4)
is being moved to the list of choice points associated with goal
(DOUBLE-INSERT PART-J IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H5 OF PART-H AND FEATURE J1 IN FEATURE H6).

Feature level subgoal (DOUBLE-INSERT FEATURE J1 OF PART-J IN FEATURE H3 OF PART-H AND FEATURE J3 IN FEATURE H4)
is being moved to the list of choice points associated with goal
(DOUBLE-INSERT PART-J IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H5 OF PART-H AND FEATURE J1 IN FEATURE H6).

Feature level subgoal (DOUBLE-INSERT FEATURE J1 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J3 IN FEATURE H7)
is being moved to the list of choice points associated with goal
(DOUBLE-INSERT PART-J IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J1 IN FEATURE H7).

Feature level subgoal (DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J1 IN FEATURE H6)
is being moved to the list of choice points associated with goal
(DOUBLE-INSERT PART-J IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J1 IN FEATURE H7).

Feature level subgoal (DOUBLE-INSERT FEATURE J1 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J3 IN FEATURE H6)
is being moved to the list of choice points associated with goal
(DOUBLE-INSERT PART-J IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J1 IN FEATURE H7).

Feature level subgoal (DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J1 IN FEATURE H5)
is being moved to the list of choice points associated with goal
(DOUBLE-INSERT PART-J IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J1 IN FEATURE H7).

Feature level subgoal (DOUBLE-INSERT FEATURE J1 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J3 IN FEATURE H5)
is being moved to the list of choice points associated with goal
(DOUBLE-INSERT PART-J IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J1 IN FEATURE H7).

Feature level subgoal (DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J1 IN FEATURE H4)
is being moved to the list of choice points associated with goal
(DOUBLE-INSERT PART-J IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J1 IN FEATURE H7).

Feature level subgoal (DOUBLE-INSERT FEATURE J1 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J3 IN FEATURE H4)
is being moved to the list of choice points associated with goal
(DOUBLE-INSERT PART-J IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J1 IN FEATURE H7).

Feature level subgoal (DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J1 IN FEATURE H3)
is being moved to the list of choice points associated with goal
(DOUBLE-INSERT PART-J IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J1 IN FEATURE H7).

Feature level subgoal (DOUBLE-INSERT FEATURE J1 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J3 IN FEATURE H3)
is being moved to the list of choice points associated with goal
(DOUBLE-INSERT PART-J IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H2 OF PART-H AND FEATURE J1 IN FEATURE H7).

Feature level subgoal (DOUBLE-INSERT FEATURE I1 OF PART-I IN FEATURE H2 OF PART-H AND FEATURE I3 IN FEATURE H6)
is being moved to the list of choice points associated with goal
(DOUBLE-INSERT PART-I IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE I1 OF PART-I IN FEATURE H2 OF PART-H AND FEATURE I3 IN FEATURE H7).

Feature level subgoal (DOUBLE-INSERT FEATURE I1 OF PART-I IN FEATURE H2 OF PART-H AND FEATURE I3 IN FEATURE H5)
is being moved to the list of choice points associated with goal

(DOUBLE-INSERT PART-I IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE I1 OF PART-I IN FEATURE H2 OF PART-H AND FEATURE I3 IN FEATURE H7).

Feature level subgoal (DOUBLE-INSERT FEATURE I1 OF PART-I IN FEATURE H2 OF PART-H AND FEATURE I3 IN FEATURE H4)
is being moved to the list of choice points associated with goal
(DOUBLE-INSERT PART-I IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE I1 OF PART-I IN FEATURE H2 OF PART-H AND FEATURE I3 IN FEATURE H7).

Feature level subgoal (DOUBLE-INSERT FEATURE I1 OF PART-I IN FEATURE H2 OF PART-H AND FEATURE I3 IN FEATURE H3)
is being moved to the list of choice points associated with goal
(DOUBLE-INSERT PART-I IN PART-H) as there is a symmetrical alternative in
(DOUBLE-INSERT FEATURE I1 OF PART-I IN FEATURE H2 OF PART-H AND FEATURE I3 IN FEATURE H7).

Heuristic find-distinguishing-feature applies to feature level subgoal:

(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H5 OF PART-H AND FEATURE K1 IN FEATURE H6).

Heuristic find-distinguishing-feature applies to feature level subgoal:

(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H5 OF PART-H AND FEATURE J1 IN FEATURE H6).

Subgoal Selection

Based on the confidence rates associated with feature level subgoals
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H5 OF PART-H AND FEATURE J1 IN FEATURE H6) has been chosen as
the most promising option for goal (DOUBLE-INSERT PART-J IN PART-H)

Based on the confidence rates associated with feature level subgoals
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H5 OF PART-H AND FEATURE K1 IN FEATURE H6) has been chosen as
the most promising option for goal (DOUBLE-INSERT PART-K IN PART-H)

Constraint Analysis

The accessibility constraints that result from the current
choice of mating features for each of the object level goals are:

Object Level Goal: (DOUBLE-INSERT PART-I IN PART-H).

Feature Level Subgoal: (DOUBLE-INSERT FEATURE I1 OF PART-I IN FEATURE H2 OF PART-H AND FEATURE I3 IN FEATURE H7)

Accessibility preconditions: (ACCESSIBLE H2) (ACCESSIBLE H7) (ACCESSIBLE I1) (ACCESSIBLE I3)

Object Level Goal: (DOUBLE-INSERT PART-J IN PART-H).

Feature Level Subgoal: (DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H5 OF PART-H AND FEATURE J1 IN FEATURE H6)

Accessibility preconditions: (ACCESSIBLE H5) (ACCESSIBLE H6) (ACCESSIBLE J3) (ACCESSIBLE J1)

Object Level Goal: (DOUBLE-INSERT PART-K IN PART-H).

Feature Level Subgoal: (DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H5 OF PART-H AND FEATURE K1 IN FEATURE H6)

Accessibility preconditions: (ACCESSIBLE H5) (ACCESSIBLE H6) (ACCESSIBLE K3) (ACCESSIBLE K1)

The accessibility condition (ACCESSIBLE H5) is propagated forward from goal
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H5 OF PART-H AND FEATURE K1 IN FEATURE H6)
and gets added to the set of postconditions of goal (DOUBLE-INSERT PART-J IN PART-H).

The accessibility condition (ACCESSIBLE H6) is propagated forward from goal
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H5 OF PART-H AND FEATURE K1 IN FEATURE H6)
and gets added to the set of postconditions of goal (DOUBLE-INSERT PART-J IN PART-H).

The planner checks to see if the execution of
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H5 OF PART-H AND FEATURE J1 IN FEATURE H6) will violate
any of the postconditions associated with (DOUBLE-INSERT PART-J IN PART-H).

Reasoning about the geometry of the features indicates that feature H5
will be eliminated by executing the subgoal
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H5 OF PART-H AND FEATURE J1 IN FEATURE H6).

An alternative feature subgoal
(DOUBLE-INSERT FEATURE J1 OF PART-J IN FEATURE H5 OF PART-H AND FEATURE J3 IN FEATURE H6)
is chosen for the object level goal (DOUBLE-INSERT PART-J IN PART-H) to avoid
the earlier conflict that arose from the constraint propagation.

```
The accessibility condition (ACCESSIBLE H5) is propagated forward from goal
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H5 OF PART-H AND FEATURE K1 IN FEATURE H6)
and gets added to the set of postconditions of goal (DOUBLE-INSERT PART-J IN PART-H).

The accessibility condition (ACCESSIBLE H6) is propagated forward from goal
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H5 OF PART-H AND FEATURE K1 IN FEATURE H6)
and gets added to the set of postconditions of goal (DOUBLE-INSERT PART-J IN PART-H).

The planner checks to see if the execution of
(DOUBLE-INSERT FEATURE J1 OF PART-J IN FEATURE H5 OF PART-H AND FEATURE J3 IN FEATURE H6) will violate
any of the postconditions associated with (DOUBLE-INSERT PART-J IN PART-H).

Reasoning about the geometry of the features indicates that feature H5
will be eliminated by executing the subgoal
(DOUBLE-INSERT FEATURE J1 OF PART-J IN FEATURE H5 OF PART-H AND FEATURE J3 IN FEATURE H6).

An alternative feature subgoal
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H4 OF PART-H AND FEATURE J1 IN FEATURE H5)
is chosen for the object level goal (DOUBLE-INSERT PART-J IN PART-H) to avoid
the earlier conflict that arose from the constraint propagation.

The accessibility condition (ACCESSIBLE H5) is propagated forward from goal
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H5 OF PART-H AND FEATURE K1 IN FEATURE H6)
and gets added to the set of postconditions of goal (DOUBLE-INSERT PART-J IN PART-H).

The planner checks to see if the execution of
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H4 OF PART-H AND FEATURE J1 IN FEATURE H5) will violate
any of the postconditions associated with (DOUBLE-INSERT PART-J IN PART-H).

Reasoning about the geometry of the features indicates that feature H5
will be eliminated by executing the subgoal
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H4 OF PART-H AND FEATURE J1 IN FEATURE H5).

An alternative feature subgoal
(DOUBLE-INSERT FEATURE J1 OF PART-J IN FEATURE H4 OF PART-H AND FEATURE J3 IN FEATURE H5)
is chosen for the object level goal (DOUBLE-INSERT PART-J IN PART-H) to avoid
the earlier conflict that arose from the constraint propagation.

The accessibility condition (ACCESSIBLE H5) is propagated forward from goal
(DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H5 OF PART-H AND FEATURE K1 IN FEATURE H6)
and gets added to the set of postconditions of goal (DOUBLE-INSERT PART-J IN PART-H).

The planner checks to see if the execution of
(DOUBLE-INSERT FEATURE J1 OF PART-J IN FEATURE H4 OF PART-H AND FEATURE J3 IN FEATURE H5) will violate
any of the postconditions associated with (DOUBLE-INSERT PART-J IN PART-H).

Reasoning about the geometry of the features indicates that feature H5
will be eliminated by executing the subgoal
(DOUBLE-INSERT FEATURE J1 OF PART-J IN FEATURE H4 OF PART-H AND FEATURE J3 IN FEATURE H5).

An alternative feature subgoal
(DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H3 OF PART-H AND FEATURE J1 IN FEATURE H4)
is chosen for the object level goal (DOUBLE-INSERT PART-J IN PART-H) to avoid
the earlier conflict that arose from the constraint propagation.


--------------------------------------------------
         Tentative Plan at the Feature Level
--------------------------------------------------


 Object Level Goal:   (DOUBLE-INSERT PART-I IN PART-H).

       Feature Level Subgoals

       (DOUBLE-INSERT FEATURE I1 OF PART-I IN FEATURE H2 OF PART-H AND FEATURE I3 IN FEATURE H7).

 Object Level Goal:   (DOUBLE-INSERT PART-J IN PART-H).

       Feature Level Subgoals

       (DOUBLE-INSERT FEATURE J3 OF PART-J IN FEATURE H3 OF PART-H AND FEATURE J1 IN FEATURE H4).

 Object Level Goal:   (DOUBLE-INSERT PART-K IN PART-H).

       Feature Level Subgoals

       (DOUBLE-INSERT FEATURE K3 OF PART-K IN FEATURE H5 OF PART-H AND FEATURE K1 IN FEATURE H6).
#<VMS4-PATHNAME "GRAFIX:RBT$DISK:[ROBOT.PLANNER]TEST2.LISP">
NIL
(dribble-end)
```