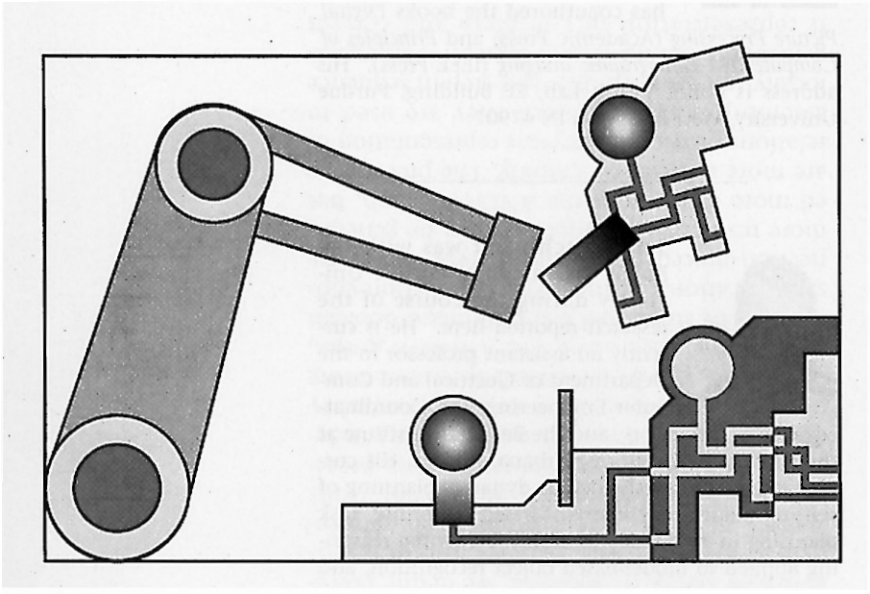
Assembly Sequence Planning

Arthur C. Sanderson, Luiz S. Homem de Mello, and Hui Zhang

The sequence of mating operations that can be carried out to assemble a group of parts is constrained by the geometric and mechanical properties of the parts, their assembled configuration, and the stability of the resulting subassemblies. An approach to representation and reasoning about these sequences is described here and leads to several alternative explicit and implicit plan representations. The Pleiades system will provide an interactive software environment for designers to evaluate alternative systems and product designs through their impact on the feasibility and complexity of the resulting assembly sequences.

Assembly plays a fundamental role in the manufacturing of most products. Parts that have been individually formed or machined to meet designed specifications are assembled into a configuration that achieves the functions of the final product or mechanism. The economic importance of assembly as a manufacturing process has led to extensive efforts to improve the efficiency and cost effectiveness of assembly operations.



In recent years, the use of programmable and flexible automation has enabled the partial or complete automation of assembly of products in smaller volumes and with more rapid product changeover and model transition. AI is increasingly playing a key role in such flexible automation systems. This article describes AI tools that facilitate reasoning about geometry, mechanics, and operations for assembly sequence planning.

In practice, manual labor, fixed automation, and flexible automation are often combined in modern manufacturing systems to take advantage of cost and reliability tradeoffs. Decisions regarding alternative assembly manufacturing technologies, tools for assembly manufacturing system design, and methods for assembly system implementation are key challenges that currently face production engineers. More systematic approaches to the analysis, design, and planning of these assembly systems are needed to enhance their performance and enable their cost-effective implementation. The work described in this article focuses on the representation of assembly sequence plans and the development of assembly sequence planning tools that form the basis for automated and interactive assembly system design methods. The Pleiades system, or planning environment for integrated assembly system design, described

0738-4602/89/$4.00 ©1990 AAAI

here, is an effort to develop such a set of software tools.

Although assembly has important applications in manufacturing, the assembly process itself has attracted scientific interest as an example of intelligent robotic manipulation. The mating of two parts with complex geometries typically requires the integration of sensory and motor control information with an internal representation of parts, geometries, and relationships. Humans carry out these manipulation tasks using well-practiced skills of integration of motor control and sensory interpretation with stored models of geometry. The replication of these skills in automated robotic systems has proven to be extremely difficult. Fundamental issues in robotics, control theory, pattern recognition, and AI are raised by this complex task. A number of generic assembly problems in manipulation—put the peg in the hole—sensing—find the part in the bin—and planning—put block A on block B—have evolved as classical challenges in the scientific literature and as means to evaluate and compare approaches and algorithms. Assembly sequence planning can also be thought of as a generic scientific problem in that the fundamental properties of assembly relations, geometries, and operations are used to guide the search for correct, complete, optimal, or desirable sequences.

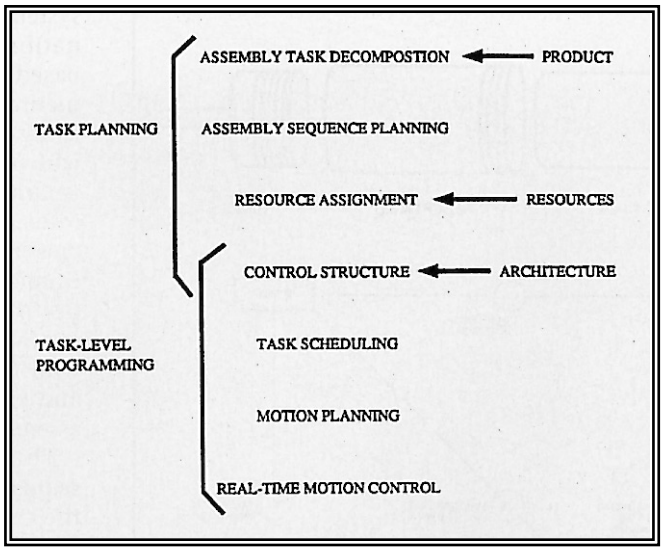


Figure 1. Hierarchy of Planning Processes for Assembly System Design and Implementation.

Assembly sequence planning is part of a hierarchy of steps in assembly system design for manual, fixed automation, or programmable automation systems.

Blocks world is a simple assembly planning environment. A goal state in blocks world specifies the contacts between a set of parts, and the PUTON(A,B) operation can be thought of as a mating operation that requires geometric access to the mating surface as well as stability of the resulting configuration. However, many of the domain-independent approaches (Fikes and Nilsson 1972; Sacerdoti 1973; Wilkins 1984; Chapman 1987; Korf 1987) to planning in blocks world do not map well into the more generalized assembly problem. The use of a propositional representation for states and subgoals, such as in Strips (Fikes and Nilsson 1972), has limitations when faced with the generalized geometries and mechanisms incorporated in product assemblies. An alternative approach to the decomposition of the planning problem is needed. Although the representation of state is more complex in the assembly planning environment than in blocks world, there are domain-specific ordering constraints for assembly that can be used to simplify the representation of plans.

In this article, we describe the use of the AND/OR graph for assembly sequence plan representation. The AND/OR graph provides a compact representation of assembly plans and is equivalent to the directed graph of assembly states. In addition, we define precedence relations that capture the domain-specific ordering constraints between connections and assembly states and connections and connections. The connection-state precedence relations (type 1) require some independence assumptions among assembly operations, can be generated by enumerating states sequences using the AND/OR graph, and can be simplified using standard Boolean simplification routines. The connection-connection precedence relations (type 2) require more restrictive assumptions, can be generated more easily from the AND/OR graph, but are more difficult to simplify. The precedence relations provide an implicit representation of assembly sequences when they are used to locally test for a feasible next step in sequence generation, but the AND/OR graph is an explicit representation of complete and correct sequences.

Assembly sequence planning is part of a hierarchy of steps in assembly system design for manual, fixed automation, or programmable automation systems. One such hierarchy of design and implementation for a

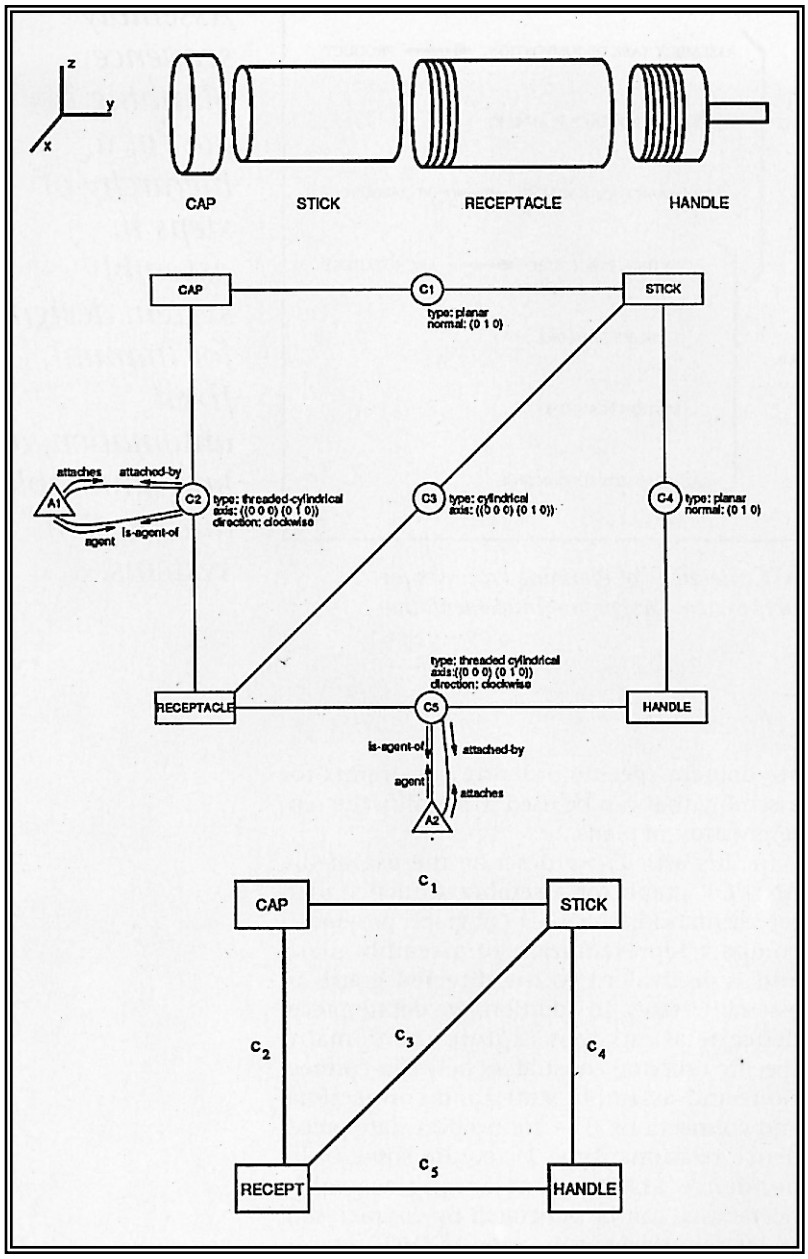


Figure 2. Three Levels of Representation of Assembly Product.

Figure 2a shows the solid model of parts, Figure 2b shows the relational model of assembly, and Figure 2c shows the graph of connections of assembly.

programmable assembly system is shown in figure 1. In this view, task-level planning of the system is carried out based on a description of the product and its parts; a description of the system and its resources, such as robots, grippers, fixtures, or sensors; and a description of the coordination architecture, including computing and communications capabilities. The resulting task-level plan describes the decomposition of the assembly task, the assignment of assembly subtasks to system resources, and a model for the coordination and scheduling of system resources based on architectural features. The implementation of the task-level plan is carried out using a task-level programming approach in which the detailed control of paths and trajectories, fine motion, grasping, sensing, and interaction forces is specified. Real-time execution utilizes this planning structure as a framework to provide efficient and reliable performance. Developing a planning and control framework that can successfully cope with the uncertainties in design specification and execution parameters is a major goal in assembly planning research.

This article is concerned with planning sequences of assembly operations that satisfy the conditions of feasibility and yield stable states. The article provides an overview of an approach and examples of results. Additional details can be found in other publications (Homem de Mello and Sanderson 1986, 1987, 1988, 1989a, 1989b, and 1990; Homem de Mello 1989; Sanderson and Homem de Mello 1989; and Zhang 1989). Assembly Sequence Planning describes the relational graph structure that we use as the basis for assembly representation and abstracts the problem of task decomposition to identify the cut sets of the graph of connections. Assembly Sequence Representation describes alternative assembly sequence representations and illustrates some advantages and properties of the AND/OR graph. Precedence Relations defines two types of precedence relations that can be generated from the AND/OR graph. These precedence relations can be simplified when independence properties of the assemblies hold; in addition, the real-time properties of precedence relations can be guaranteed under certain conditions. Evaluation and Selection of Assembly Plans discusses evaluation functions for assembly sequence plans. The Pleiades System summarizes the system, which is an approach to developing an environment for assembly system planning. The closing section presents conclusions and directions for continuing work.

Assembly Sequence Planning

An assembly product description consists of the geometric description of each of the individual parts, their geometric tolerances, and the configuration in which the parts fit together to form the final product. In practice, such a description is often incomplete or inexact and strongly relies on human experience and intuition for its full interpretation. For our purposes here, we assume that an

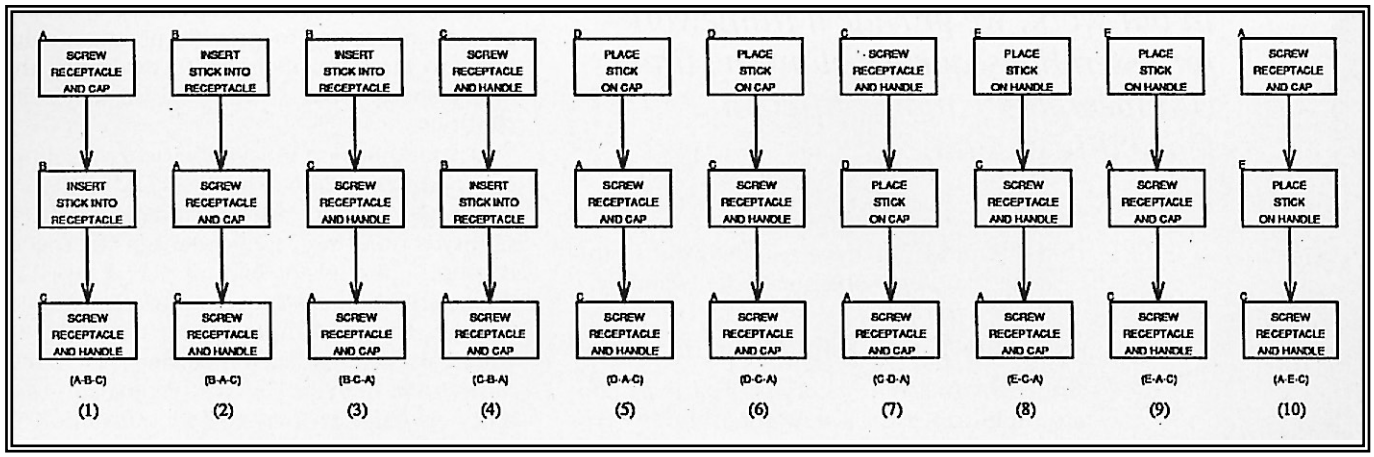


Figure 3. Feasible Assembly Sequences for the Product Shown in Figure 2.

explicit description of the parts geometry and the assembled configurations is given or derived. Our approach to assembly representation is strongly influenced by previous experience with relational models (Lieberman and Wesley 1977; Eastman 1981; Lee and Gossard 1985). We do not currently consider tolerances for planning purposes, although the geometric modeling system we use incorporates them (see The Pleiades System) and provides access for use in the planning procedures. In addition to the geometric configuration, most assemblies use attachments that apply physical forces to constrain the relative motion among parts and stabilize the final assembly. Although attachments such as screws and clips are geometrically represented in the assembly description, it is often impossible to infer from their geometric description what the physical role of these parts might be as an attachment for the assembly. Therefore, we interactively add the attachment description to the relational model.

In our work, three levels of description are used for assembly product representation. An example for a simple product is shown in figure 2. First, as figure 2a illustrates, is a complete computer-aided description (CAD) of individual parts geometries as output by the Catia (IBM Corp.) system. Second, from the CAD-based description of parts, we derive a relational graph model of the resulting assembly, as shown in figure 2b. This relational graph extraction is based on an object-oriented geometric modeling system, Geos (see The Pleiades System), which provides an explicit representation of parts and contacting surfaces in the assembly. We interactively add attachment entities to explicitly describe constraints on the degrees of freedom of contacts within the structure. In this description, each attachment has an agent and a contact, where each contact has its degrees of freedom constrained when the attachment agent is present. These relationships directly influence the feasible assembly sequences because an attachment agent must often be removed prior to removing a part that breaks a contact. Third is a simplified relational structure called the graph of connections, as shown in figure 2c. Although the relational graph contains specific properties and attributes associated with each of the entities, the graph of connections is the relational structure that defines the parts and connections. The three levels of product description shown in figure 2 are all necessary for assembly sequence planning, and this hierarchical organization facilitates the planning tasks.

Ten different feasible sequences can be used to assemble the product shown in figure 2. These sequences are listed in figure 3. From a planning perspective, each of these sequences is a series of actions that satisfies the preconditions for each action and that leads to the goal state. In assembly sequence planning, the *initial state* is always the state in which no parts are interconnected, and the *goal state* is always the state in which all parts are interconnected in the final unique configuration. The intermediate system states consist of subsets of mutually interconnected parts called *subassemblies.* We assume for this discussion that the geometric configuration of a subassembly is uniquely specified by its constituent parts; that is, there is only one way for a given subset of parts to fit together. For these examples, therefore, if we assemble a cylinder with a piston, we uniquely specify the final position of the piston and assume

In our work, we provide a framework for assembly sequence planning that can incorporate many different feasibility criteria.

that it would not have to occupy multiple discrete states to enable successful assembly.

Assembly Task Feasibility Predicates

An *assembly action* mates two parts or subassemblies to form a new subassembly. (For the current discussion, we assume that only two—and not more—subassemblies are joined at a given time.) In assembly, the preconditions for such an action can be thought of as the feasibility conditions for the execution of the operation and the acceptability of the resulting state. In this work, we introduce a hierarchy of such feasibility conditions to reduce the complexity of the geometric and physical reasoning that must be carried out for sequence generation. In practice, the number of sequences that must be examined grows exponentially with the number of parts, and the use of detailed geometric or physical analysis at each step is computationally unrealistic. We are developing a hierarchy of feasibility criteria that invoke computationally efficient necessary conditions early in the search to prune the tree of possible sequences.

For example, in the following discussion, local geometric feasibility is a subset of global geometric feasibility and, therefore, is unnecessary to compute as a separate criterion. However, local geometric feasibility can be efficiently computed using only the contact normal information from the relational model and does not require full geometric shapes of parts or general part-shape analysis. Local geometric feasibility is a necessary but not sufficient condition for a feasible sequence and reduces the overall complexity of the analysis. Similarly, resource-independent feasibility is necessary but not sufficient with respect to the availability of tools for a given operation. The benefits of these hierarchies depend on individual cases, but the approach is designed to provide a practical approach to contend with the high combinatorial complexity that arises in real assembly problems. The hierarchies we describe here are analytically based; that is, they are based on geometric and physical relations rather than heuristics. This approach emphasizes the use of analytic methods to gain generality for a class of problems. In practice, heuristics can be appropriate and necessary to provide practical solutions to large problems or to deal with the many specific cases that arise in assembly relations.

Preconditions on the feasibility of operations can be broken down into several different categories: First are resource-independent feasibility conditions: Independent of the specific robots, grippers, or fixtures that are used in the assembly, there are geometric constraints among the parts that restrict the order in which operations can be carried out. These restrictions include (1) local geometric feasibility (Is local or incremental translation of the part feasible from its final assembled state? This question can be answered for translational motion using only surface-normal information that is stored in the relational graph [Homem de Mello 1989].) and (2) global geometric feasibility (Is there an unobstructed path from some remote position to the final assembled state given the geometric obstructions posed by other parts that are present in the assembly state? This problem is related to the general path-planning problem [Lozano-Perez and Wesley 1979] but only requires determination of the existence of a path and not necessarily the specification of the path.).

Second are resource-dependent feasibility conditions: (1) geometric feasibility (Is there an unobstructed path for mating of parts given the geometry of the parts plus the geometry of robots, grippers, and fixtures that can be used in the operation?), (2) attachment feasibility (Can appropriate forces be exerted by available tools to attach parts as required by the mating operation?), and (3) tool availability (Is there an appropriate robot, gripper, or fixture available to carry out a particular task to meet geometric and attachment constraints?).

Although such feasibility conditions might be examined simultaneously, it is often more efficient to hierarchically structure the planning approach. As suggested by figure 1, we can examine assembly sequence plans that are feasible from a resource-independent point of view, and these plans subsume all the feasible plans that incorporate resource dependencies. The resource-independent plan reduces the search space for the resource-dependent planning. In addition, a number of heuristics related to the complexity of operations and the desirability of states can be added at this stage to further reduce the search space. Figure 3 shows a set of assembly sequences for the simplified flashlight product that are all feasible from a resource-independent criterion. An example of an

infeasible sequence for this product would be sequence A-C-B. In this sequence, the cap and handle are attached to the receptacle before the stick is placed inside. It becomes geometrically infeasible to insert the stick into the receptacle when both ends are attached. Such a sequence should be rejected early in the planning process and not considered further. Once such a sequence has been rejected based on local geometric feasibility conditions, it is unnecessary to examine global geometric feasibility or resource-dependent feasibility conditions. Another sequence such as B-A-C might be geometrically feasible but undesirable. In this case, the stick is inserted into the receptacle resulting in a subassembly state that might require fixturing to keep the stick from sliding out. Such a state might be detected and evaluated as undesirable by an appropriate heuristic. Resource-dependent feasibility constraints could further limit the set of sequences shown in figure 3. If there were no gripper available to pick up the stick or no mechanism available to screw in the handle, then there might be no remaining feasible sequences.

In addition to task feasibility conditions, we consider state feasibility conditions. Again, several resource-independent and resource-dependent conditions can come into play. These include (1) stability (Is there one pose of the subassembly for which the configuration is stable? This problem is difficult in general [Bonenschanscher 1988] and is strongly influenced by the types of friction and surface-interaction assumptions that are made.) and (2) rigidity (Is the subassembly stable for all poses of the subassembly?). Each of these criteria can be examined under different assumptions: (1) no gravity or external forces —in this case, all subassemblies are stable and rigid—and (2) gravity or other external forces —in this case, stability can strongly depend on the orientation of the subassembly, and rigidity often requires attaching all parts. Fixtures or grippers can be added to the configuration to impose stability or rigidity on a subassembly during the assembly process. All the assembly sequences presented in figure 3 have at least one stable pose for each state in the presence of gravity, although the pose might be changed during the sequence to ensure stability and enable geometric feasibility. In general, the assessment of the stability of a state would also require a description of the frictional forces that arise.

Assembly Sequence Planning

In our work, we provide a framework for assembly sequence planning that can incorporate many different feasibility criteria. The same approach to plan generation, plan representation, and plan evaluation can be used for different feasibility conditions. As discussed earlier, the resource-independent plans subsume the resource-dependent plans and, therefore, simplify the overall planning process. In the experiments described here, we implemented two feasibility criteria for the generation of assembly sequences: (1) resource-independent local geometric feasibility and (2) attachment feasibility based on the assumption that attachment tools are available and that breaking attached contacts is infeasible unless the attachment agent is removed. These two feasibility criteria are sufficiently interesting and complex to illustrate many of the important problems that occur in assembly sequence planning.

The problem of generating feasible assembly sequences for a product can be transformed into the problem of generating disassembly sequences for the same product. For many problems of interest, this transformation reduces the branching factor of the search space because often many more options and deadends occur in assembly than in disassembly. This transformation of the problem is conceptual rather than physical. Assembly tasks are not necessarily reversible in the physical sense, but we impose this reversibility from a conceptual standpoint to plan correct sequences. This decomposition approach to assembly planning maps nicely onto our relational model of the product and leads to a recursive decomposition of the task in which each decomposable subproblem is a proper subset of the original model. In addition, as we show in the next section, this decomposition approach lends itself to the AND/OR graph representation of assembly sequences.

The basic planning strategy is to recursively enumerate the decompositions of the assembly and to select the decompositions that are feasible by imposing a set of feasibility predicates. The decompositions of the assembly are enumerated by the cut sets of the assembly's graph of connections. The feasible operations and feasible state predicates are evaluated using the relational model and the geometric parts models.

Assembly States

The assembly graph of connections can be represented by a simple undirected graph $\langle P, C \rangle$ in which the parts $P = \{p_1, p_2, \ldots p_N\}$ are the set of nodes, and the connections $C = \{c_1, c_2, \ldots, c_L\}$ are the set of edges. A *state* of the assembly process is a configuration of the parts at the beginning or end of an assembly

task. The state of an assembly process can be characterized either by the configuration of contacts that have been established or the partitions of the parts that are connected. In the first case, a state of the assembly process can be represented by an L-dimensional binary vector $\mathbf{x} = [x_1, x_2, \ldots, x_L]$ in which the i^{th} component of x_i is true if the i^{th} connection is established in this state. For example, if the first task of the assembly process for the example shown in figure 2 is the joining of the cap to the receptacle, the second state of the assembly process can be represented by [false, true, false, false, false].

Alternatively, an assembly state can be characterized by partitioning of parts into subassemblies. For example, if the first task is the joining of the cap to the receptacle, the second state of the assembly process can be represented by { {CAP, RECEPTACLE}, {STICK}, {HANDLE} }. Given an assembly's graph of connections and one of the two representations of the assembly state, it is straightforward to obtain the other representation. It is also important to observe that not all partitions and not all L-dimensional binary vectors can characterize a proper assembly state. For example, for the assembly shown in figure 2, the five-dimensional binary vector [true, true, false, false, false] does not correspond to a state because if connections c_1 and c_2 are established, then c_3 must also be established. A subassembly predicate *sa* is defined to determine whether a subset of parts makes up a subassembly and, therefore, whether a given binary vector is, in fact, a proper state vector of the assembly. In addition, we define a subassembly's stability predicate *st*, which determines whether a subassembly described by its set of parts and connections is stable. Although we have studied a subassembly's stability predicate based on the stability of parts configuration subject to gravity and zero friction, many other assumptions and analysis tools or heuristics might be implemented here. An assembly-state representation in which all subassemblies satisfy the stability predicate is said to be a stable assembly-state representation.

Assembly Tasks

Given two subassemblies characterized by their sets of parts θ_i and θ_j, we say that joining θ_i and θ_j is an assembly task if the set $\theta_k = \theta_i \cup \theta_j$ characterizes a subassembly. Equivalently, an assembly task can be characterized by the output subassembly and the set of connections that are established by the task. For such a set of connections to represent an assembly task, it must correspond to a cut set of the graph of connections of the task's output subassembly. Conversely, each cut set of a subassembly's graph of connections corresponds to an assembly task.

Each assembly task must be evaluated by the operations feasibility predicates described previously. An assembly task is said to be resource-independent, geometrically feasible if there is a collision-free path to bring the two subassemblies into contact from a situation in which they are far apart. We further decompose this geometric feasibility predicate *gf* into a local predicate and a global predicate. The local *gf* predicate tests for the incremental motion of the designated subassemblies, and the global *gf* predicate tests for the existence of a global collision-free path. An assembly task is said to have attachment feasibility if it is feasible to establish the attachments that act on the contacts between the two subassemblies. The attachment feasibility predicate *af* evaluates this predicate based on the attachment description that is incorporated into the relational model. A local geometric feasibility can also be evaluated using the contact attributes and properties established in the relational model. Global geometric feasibility requires access to the detailed parts descriptions that are referenced from the relational model to the parts models. The computation of these global *gf* predicates can be approached in a variety of different ways. They are complex to compute in general and incorporate many challenging and interesting problems in geometric and physical reasoning. As mentioned earlier, to demonstrate the overall planning framework, the experiments described in this article include a local geometric feasibility predicate and an attachment feasibility predicate.

Our algorithm for the generation of assembly sequences uses an approach that enumerates the decompositions of the assembly and then selects those decompositions that are feasible. As shown in the next section, this recursive decomposition results in the construction of the AND/OR graph representation of assembly plans. The procedure GET-FEASIBLE-DECOMPOSITIONS takes as input the relational model of an assembly and returns all feasible decompositions of this assembly. The procedure first generates the graph of connections for the input assembly and computes the cut sets of this graph. The cut sets are enumerated by looking at all connected subgraphs having the cardinality of their set of nodes smaller than or equal to half of the cardinality of the set of nodes in the whole graph. For each of these subgraphs,

the set of edges of the whole graph that have only one end in the subgraph defines a cut set if their removal leaves the whole graph with exactly two components. Each resulting cut set corresponds to a decomposition of the graph. The procedure GET-DECOMPOSITION finds this decomposition, and the procedure FEASIBILITY-TEST is used to check whether this decomposition is feasible. The procedure FEASIBILITY-TEST further breaks down into a set of procedures that implements the individual operation feasibility predicates and state feasibility predicates.

The complexity of the generation of assembly sequences depends on the number of parts, *N*, and the interconnections among the parts. The number of possible decompositions, *D* (that is, cut sets of the graphs of connections), can be used as an upper bound on the complexity of sequence generation. In practice, sequences are not generated exhaustively, rather the search tree for desirable sequences can be pruned during the generation process. We examined complexity for a strongly connected assembly, where every part is connected to every other part, and a weakly connected assembly, where there are *N-1* connections among the *N* parts.

For a one-part-at-a-time decomposition, where each cut set results in at least one one-part subassembly, the order of complexity for a strongly connected interconnection is $D = O(2^N)$ and for a weakly connected interconnection is $D = O(N^2)$. For a network decomposition, where all cut sets are feasible decompositions, the order of complexity for a strongly connected interconnection is $D = O(3^N)$ and for a weakly connected interconnection is $D = O(N^3)$.

The relative efficiency of the representation of the resulting plans depends on the nature of the plan representation that is used. These complexities are based on the use of the AND/OR graph assembly sequence representation that is described in the next section.

Assembly Sequence Representation

As described previously, each assembly can have many different feasible assembly sequences. Our first objective is to generate these sequences and represent them in an efficient manner. Given an assembly that has *N* parts, an ordered set of *N-1* assembly tasks $\tau_1\ \tau_2 \ldots \tau_{N-1}$ is an assembly sequence if there are no two tasks that have a common input subassembly, the output subassembly of the last task is the whole assembly, and the input subassemblies to any task are either a one-part subassembly or the output subassembly of a preceding task. Such an assembly sequence can also be characterized by an ordered sequence of states in which the state s_1 is the state in which all parts are separated, the state s_N is the state in which all parts are joined forming the whole assembly, any two consecutive states are such that only the two input subassemblies of the task are in s_i and not in s_{i+1}, and only the output subassembly of task τ_i is in s_{i+1} and not in i. An assembly sequence is said to be feasible if all its assembly tasks and assembly states are feasible.

. . . each assembly can have many different feasible assembly sequences.

An assembly sequence, therefore, can be represented in several different ways: (1) an ordered list of task representations, (2) an ordered list of binary vectors, (3) an ordered list of partitions of the set of parts, or (4) an ordered list of subsets of connections. For example, a feasible assembly sequence for the product shown in figure 2 could be represented as one of the following:

- A three-element list of task representations:
  ({{CAP},{RECEPTACLE}}
  {{CAP,RECEPTACLE},{STICK}}
  {{CAP, RECEPTACLE, STICK},{HANDLE}})
- A four-element list of five-dimensional binary vectors:
  ([false,false,false,false,false]
  [false,true,false,false,false]
  [true,true,true,false,false]
  [true,true,true,true,true])
- A four-element list of partitions of the set of parts:
  ({{CAP},{RECEPTACLE},{STICK},{HANDLE}}
  {{CAP,RECEPTACLE},{STICK},{HANDLE}}
  {{CAP,RECEPTACLE,STICK},{HANDLE}}
  {{CAP,RECEPTACLE,STICK,HANDLE}})
- A three-element list of sets of connections:
  ($\{c_2\}\{c_1, c_3\}\{c_4,c_5\}$) .

Because each assembly sequence can be represented by ordered lists, it is possible to represent the set of all assembly sequences by a set of lists. Although this set of lists might represent a complete and correct description of all feasible assembly sequences, it is not necessarily the most compact or most useful representation of these sequences. In particular, because many assembly sequences share common subsequences and common states, attempts have been made to create more compact representations that can encompass all feasible assembly sequences. We introduced the AND/OR graph representation of assembly sequences as a basis for these representations. We developed an algorithm for the generation of the AND/OR graph and subsequent algorithms that generate equivalent, complete, and correct representations of the directed graph and precedence relations.

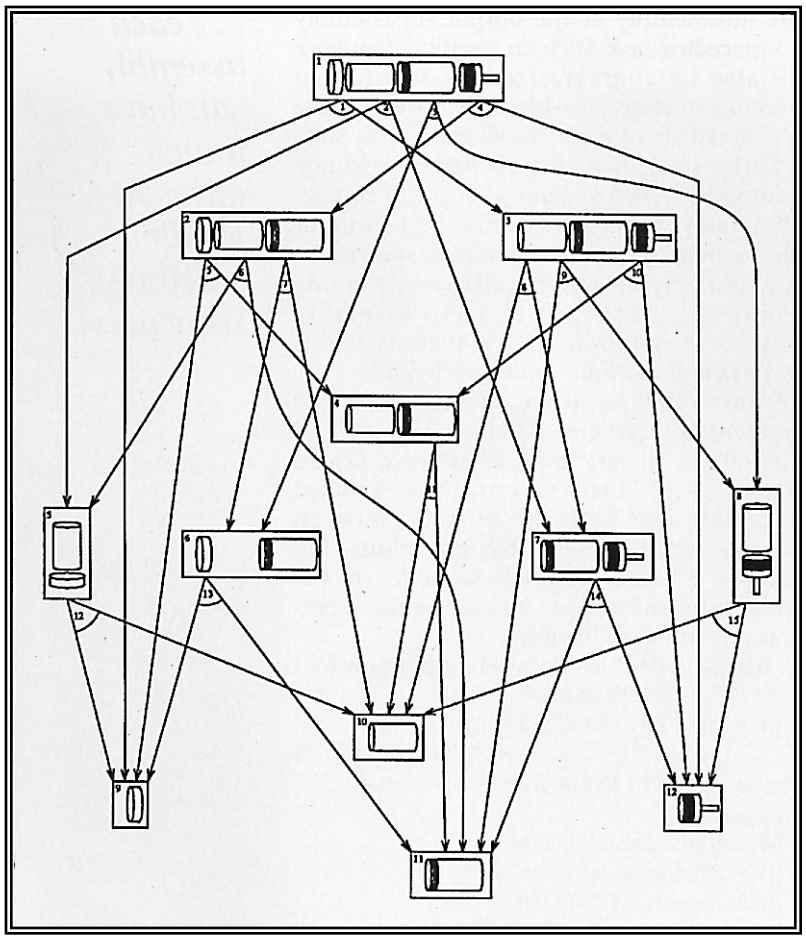


Figure 4. AND/OR Graph of the Product Shown in Figure 2.

The nodes in this AND/OR graph representation of assembly sequences are the subsets of P that characterize stable subassemblies. The hyperarcs correspond to the feasible assembly tasks. Each *hyperarc* is an ordered pair in which the first element is a node that corresponds to a stable subassembly θ_k, the second element is a set of two nodes $\{\theta_i,\theta_j\}$ such that $\theta_k = \theta_i \cup \theta_j$, and the assembly task characterized by θ_i and θ_j is feasible. Each hyperarc is associated with a decomposition of the subassembly that corresponds to its first element and can also be characterized by this subassembly and the subset of all its connections that are not in the graphs of connections of the subassemblies in the hyperarc's second element. This subset of connections associated to a hyperarc corresponds to a cut set in the graph of connections of the subassembly in the hyperarc's first element. This AND/OR graph can be formally defined as follows:

Definition: The AND/OR graph of feasible assembly sequences of an assembly whose set of parts is $P = \{p_1, p_2, \ldots, p_N\}$ is the AND/OR graph $\langle \sigma_p, D_p \rangle$ in which

$$\sigma_p = \{\theta \in \Pi(P) \mid sa(\theta) \wedge st(\theta)\}$$

is the set of stable subassemblies, and

$$D_p = \{(\theta_k,\{\theta_i,\theta_j\}) \mid [\theta_i,\theta_j,\theta_k \in \sigma_p] \wedge [U(\{\theta_i,\theta_j\}) = \theta_k] \wedge [af(\{\theta_i,\theta_j\})] \wedge [gf(\{\theta_i,\theta_j\})]\}$$

is the set of feasible assembly tasks. The notation $\Pi(P)$ is used to represent the set of all subsets of P.

As an example, figure 4 shows the AND/OR graph for the feasible sequences for the assembly shown in figure 2. This AND/OR graph representation is complete and correct in that it includes all possible feasible assembly sequences and does not include any infeasible assembly sequences. A given *assembly sequence* can be defined as a feasible assembly tree within the AND/OR graph.

The algorithm GENERATE-AND/OR-GRAPH takes the relational model of an assembly and returns the AND/OR graph representation of all assembly sequences for this assembly. The nodes in the AND/OR graph returned are pointers to relational models of subassemblies. The algorithm is not reproduced here because of space limitations, but it uses the procedure GET-DECOMPOSITIONS to generate all decompositions of the relational models and keeps track of lists of pointers to determine whether specific subassemblies have previously been generated. These procedures can be made more efficient by linking the evaluation of feasibility tests between different decompositions to avoid duplicating the computational work. An analysis of the completeness, correctness, and complexity of the AND/OR graph generation algorithm is discussed in Homem de Mello and Sanderson (1990).

Given an assembly whose graph of connections is $\langle P,C \rangle$, a directed graph can also be used to represent the set of all feasible assembly sequences. The nodes in this directed graph correspond to stable state partitions of the set P. These are the partitions Θ of P such that if $\theta \in \Theta$, then θ is a stable subassembly of P. The edges in this directed graph are ordered pairs of nodes. For any edge, there are only two subsets, θ_i and θ_j, in the state partition corresponding to the first node that are not in the state node corresponding to the second node. Therefore, each edge corresponds to an assembly task. If all assembly tasks are feasible, then the graph is referred to as the directed graph of feasible assembly sequences. Figure 5 shows the directed graph of feasible assembly sequences for the assembly shown in figure 2. A path in the directed graph of

feasible assembly sequences corresponds to a feasible assembly sequence for the assembly *P*. In such a path, the ordered sequence of edges corresponds to the ordered sequence of tasks, and the ordered sequence of nodes corresponds to the ordered sequence of assembly process states.

The equivalence of the directed graph and the AND/OR graph as complete and correct representations of feasible assembly sequences has been proven elsewhere (Homem de Mello and Sanderson 1990). The relative complexity of the two representations in terms of the number of nodes in the graphs has also been addressed in Homem de Mello and Sanderson (1990). In this analysis, for strongly connected assemblies, the AND/OR graph had fewer nodes for all assemblies larger than $N = 4$, and at $N = 10$, it had fewer nodes by more than two orders of magnitude. Although the list of sequences, the directed graph, and the AND/OR graph all represent complete sets of sequences, the AND/OR graph has the advantage in the efficiency of its representation. This efficiency can be viewed as a space-time trade-off, such that given a fixed amount of space allocated for representation storage, the AND/OR graph will tend to represent more feasible sequences and, therefore, can lead to a more optimal performance and a solution in shorter time. An example of the use of the AND/OR graph representation for the opportunistic scheduling of robotic assembly tasks is shown in Homem de Mello and Sanderson (1990).

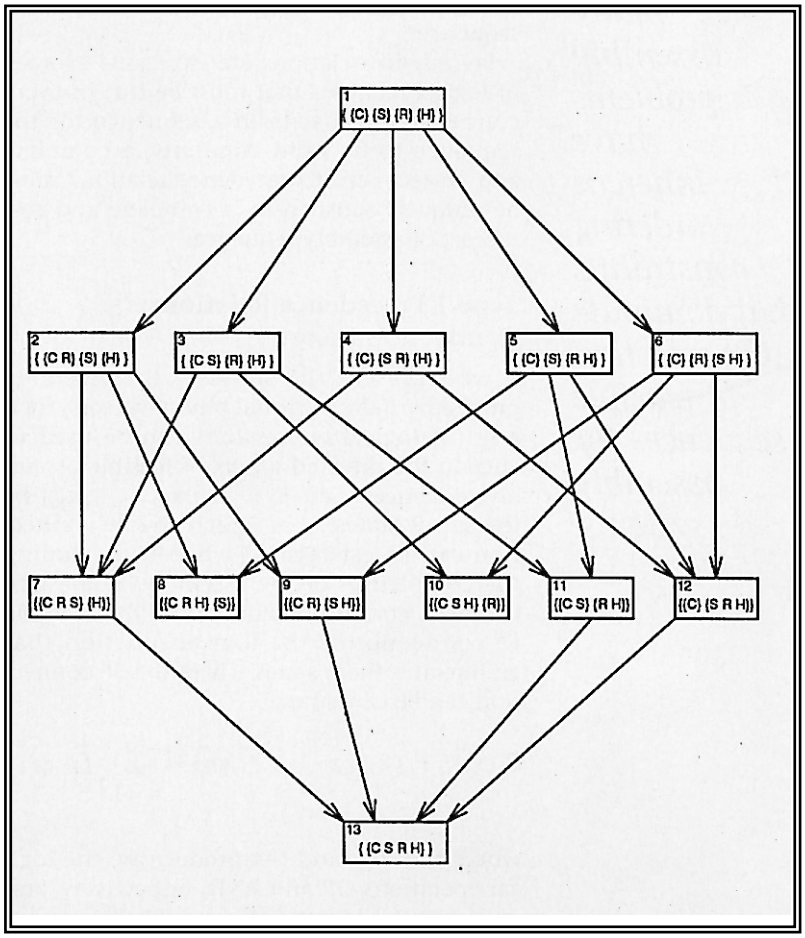


Figure 5. Directed Graph of the Product Shown in Figure 2.

Precedence Relations

It has been observed that many assembly problems have inherent ordering constraints that dominate the selection of feasible sequences for assembly systems. For the example in figure 2, "The stick must be in the receptacle before both ends are attached" is a generalization on ordering, which, in itself, is sufficient to distinguish all the feasible and infeasible sequences. Intuitive precedence relations of this type have been used by assembly designers for many years. The work of Bourjault (1984), DeFazio and Whitney (1988), and Lui (1988) has attempted to capture this intuitive knowledge by formalized sets of interactive questions provided to the designer. In our work, we have shown how to derive these sets of precedence relations directly from the AND/OR graph and, therefore, to be able to automatically generate precedence relations from the design description.

In this section, we define two types of precedence relations: type 1—precedence relations between the establishment of one connection and states of the assembly process—and type 2—precedence relations between the establishment of one connection and the establishment of another connection of the assembly process. Both types result in logical expressions on the occurrence of connections or states. They can both be generated from the AND/OR graph and are shown to be complete and correct descriptions of feasible assembly sequences. We also introduce independence properties of assemblies that permit the simplification of these precedence relations. Type 1 precedence relations require fewer assumptions than type 2 and are easier to simplify but are more difficult to generate. Type 2 precedence relations require more restrictive assumptions and are more difficult to simplify but are easier to generate from the AND/OR graph. The independence properties used in these derivations provide interesting insight into the characteristics of assemblies

. . . many assembly problems have inherent ordering constraints that dominate the selection of feasible sequences for assembly systems.

and the complexity of their resulting sequences.

Precedence relations are expressed as a set of logical relations that must be true of every connection and state in a sequence for the sequence to be valid. Similarly, a complete and correct set of precedence relations must be uniquely satisfied by a complete and correct set of assembly sequences.

Type 1 Precedence Relations: Connection-State

If we represent the states of the assembly process by L-dimensional binary vectors, then a set of logical expressions can be used to encode the directed graph of feasible assembly sequences. Let $\Xi_i = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{Ki}\}$ be the set of states from which the i^{th} connection can be established without precluding the completion of the assembly. The establishment condition (Bourjault 1984) for the i^{th} connection is the logical function that enumerates these states where the i^{th} connection can be carried out:

$$F_i(x) = F_i(x_1, x_2, \ldots, x_L) = \sum_{k=1}^{K_i} \prod_{l=1}^{L} \gamma_{kl},$$

where the sum and the product are the logical operations OR and AND, respectively, and γ_{kl} is either the symbol x_l if the l^{th} component of $\boldsymbol{x_k}$ is true or the negation $\bar{x}_l$ if the l^{th} component of $\boldsymbol{x}_k$ is false. Clearly, every element $\boldsymbol{x_k}$ of $\Xi_i = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_{Ki}}\}$ is such that $F_i(\boldsymbol{x_k})$ = true. If Ξ_i is a complete set, then any occurrence of the l^{th} connection in a feasible sequence must be preceded by a state for which $F_i(\boldsymbol{x_k})$ = true. It is often possible to simplify the expression of $F_i(\boldsymbol{x_k})$ using the rules of Boolean algebra. The set of establishment conditions defines a correct and complete set of assembly sequences, and this set of conditions can be obtained directly from the AND/OR graph, as described in Zhang (1989). The establishment conditions do not, in themselves, correspond to a complete and correct set of precedence relations. A complementary set of conditions for the infeasible assembly states can be used as a basis for deriving the type 1 precedence relations described in the following paragraphs.

We use the notation $C_i \rightarrow \Lambda(\boldsymbol{x})$ to indicate that the establishment of the i^{th} connection must precede any state s of the assembly process for which the value of the logical function $\Lambda(\boldsymbol{x})$ is true. The argument of $\Lambda(\boldsymbol{x})$ is the L-dimensional binary vector representation of the state s. We use a compact notation for logical combinations of precedence relations. For example, we write $c_i + c_j \rightarrow \Lambda(\boldsymbol{x})$ when we mean $[\boldsymbol{c_i} \rightarrow \Lambda(\boldsymbol{x})] \vee [c_j \rightarrow \Lambda(\boldsymbol{x})]$. An assembly sequence whose representation as an ordered sequence of binary vectors is $(\boldsymbol{x_1}\ \boldsymbol{x_2}\ \ldots\ \boldsymbol{x_N})$ and whose representation as an ordered sequence of subsets of connections is $(\gamma_1\ \gamma_2 \ldots \gamma_{N-1})$ satisfies the precedence relationship $c_i \rightarrow \Lambda(\boldsymbol{x})$ if

$$\Lambda(\boldsymbol{x_k}) \Rightarrow \exists\, l\, [(l < k) \wedge (c_i \in \gamma_l)]$$
$$\text{for } k = 1, 2, \ldots, N\ .$$

Let Ψ_S be the set of assembly states that never occur in any feasible assembly sequence. These states include the unstable assembly states plus stable states from which the final state cannot be reached plus the states that cannot be reached from the initial state. Let $\Psi_X = \{\boldsymbol{x}_1, \boldsymbol{x}_2 \ldots, \boldsymbol{x}_J\}$ be the set of all L-dimensional binary vectors that represent the assembly states in Ψ_S. Every element $\boldsymbol{x}_j$ of Ψ_X is such that the value of the logical function $G(\boldsymbol{x}_J)$ is true, where

$$G(\mathbf{x}) = G(x_1, x_2, \ldots, x_L) = \sum_{k=1}^{I} \prod_{l=1}^{L} \lambda_{kl}\ .$$

The sum and the product in this equation are the logical operations OR and AND, respectively, and λ_{kl} is either the symbol x_l if the l^{th} component of $\boldsymbol{x_k}$ is true or the symbol $\bar{x}_l$ if the l^{th} component of $\boldsymbol{x}_k$ is false. In many cases, the expression $G(\boldsymbol{x})$ can be simplified using the rules of Boolean algebra. Allowing for simplifications but keeping the logical function as a sum of products, you can rewrite this equation as

$$G(x) = \sum_{j=1}^{J'} g_j(x)\ ,$$

where each term $g_j(\boldsymbol{x})$ is the product of a subset of $\{x_1, x_2, \ldots x_L, \bar{x}_1, \bar{x}_2, \ldots \bar{x}_L\}$ that does not include both x_i and $\bar{x}_i$ for any i. Each term $g_j(\boldsymbol{x})$ can be further rewritten, grouping all the nonnegated variables first and all the negated variables last, for example, $g_j(\boldsymbol{x}) = x_a \cdot x_b \cdot \ldots \cdot x_h \cdot \bar{x}_p \cdot \bar{x}_q \cdot \ldots \cdot \bar{x}_z$.

Any assembly sequence that includes a state that is in Ψ_S is an unfeasible assembly sequence. Therefore, a necessary condition for the feasibility of an assembly sequence whose representation as an ordered list of binary vectors is $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_N)$ is $G(\boldsymbol{x}_1) = G(\boldsymbol{x}_2) = \ldots = G(\boldsymbol{x}_N)$ = false.

This condition is equivalent to $g_j(\boldsymbol{x}_i)$ = false for $i = 1,2, \ldots, N$ and $j = 1,2, \ldots, J'$. This necessary condition is also sufficient if the assembly has the following property:

Property 1: Given any two feasible states s_i and s_j, not necessarily in the same assembly

sequence, let γ_i and γ_j be the sets of connections that are established in assembly tasks τ_i and τ_j from s_i and s_j, respectively. If

$\langle P, C_i \rangle$ is the state's graph of connections associated to s_i,

$\langle P, C_j \rangle$ is the state's graph of connections associated to state s_j,

$$\gamma_i \subseteq \gamma_j,$$

$$C_i \subseteq C_j,$$

and

τ_j is geometrically and mechanically feasible, then τ_i is geometrically and mechanically feasible.

This property corresponds to the fact that if it is feasible to establish a set of connections when many other connections have already been established, then it is also feasible to establish fewer connections when fewer other connections have been established. Although many common assemblies have this property, there are assemblies that don't have it. Property 1 can be viewed as an independence property among connections. It assumes that no mechanisms are present in the assembly, such that when additional parts are added, the geometric access for assembly of parts is improved. Note that property 1 does not guarantee that the resulting state will be stable. In type 1 precedence relations, the state feasibility is explicitly checked by enumerating unfeasible states, and therefore, no prior assumptions are required. For type 2 precedence relations, an additional independence property is required to guarantee state feasibility of the resulting sequences. An example of an assembly that does not have property 1 is shown in figure 6.

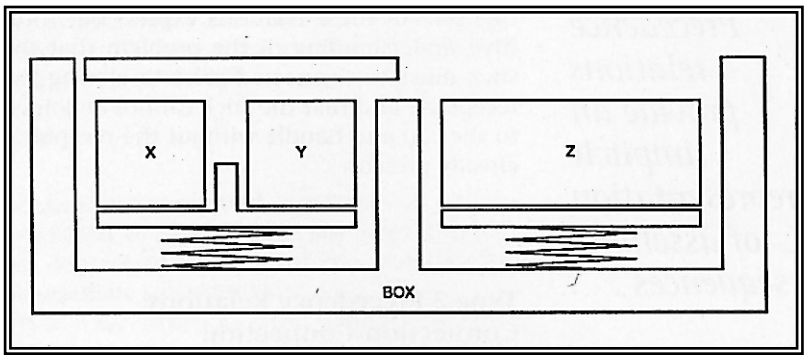


Figure 6. Example of an Assembly That Violates Property 1.

The cover of the box is a sliding mechanism, and the blocks are placed on springs. When either X *or* Y*—but not both—is present, the spring is not completely compressed, the other block prevents the cover from sliding, and* Z *cannot be inserted. Adding the other block to the assembly compresses the spring, permits the cover to slide open, and* Z *can be inserted. Because inserting* Z *is feasible with* X *and* Y *present —but not with only* X *or* Y *present individually—the assembly violates property 1.*

Homem de Mello and Sanderson (1989b) and Homem de Mello (1989) show that if $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_N)$ is an ordered list of binary vectors that represents an assembly sequence, the condition $g_j(\boldsymbol{x}_i)$ = false, $i = 1,2, \ldots, N$ is a requirement for a feasible assembly sequence and also corresponds to a precedence relationship:

$$c_p + c_q + \ldots + c_z \rightarrow S(\boldsymbol{x}) \ ,$$

where

$$S(x) = \prod_{l=1}^{L} \lambda_l = \left\{ \begin{matrix} x_l \text{ if } \lambda \in \{a, b, \ldots, h\} \\ \text{true otherwise} \end{matrix} \right\} ,$$

where

$$\{a, b, \ldots, h\} \cap \{p, q, \ldots, z\} = 0 ,$$

$$\{a, b, \ldots, h\} \cup \{p, q, \ldots, z\} \subseteq \{1, 2, \ldots, L\} \ .$$

By applying this result to each of the J' terms on the right side of the equation, we obtain J' precedence relationships. Given an assembly sequence, if it satisfies all J' precedence relationships, then it does not include any state in Ψ_S and, therefore, is feasible. Conversely, if the assembly sequence does not include any state in Ψ_S, and therefore, it is a feasible assembly sequence, then it satisfies all precedence relationships. Therefore, the set of J' precedence relationships is a correct and complete representation of the set of all feasible assembly states leading to the validation of all correct and complete sequences. This result is proven as a theorem in Homem de Mello and Sanderson (1989b) and Zhang (1989). The following example illustrates the application of this result. An algorithm for the generation of these type 1 precedence relations from the AND/OR graph is given in Zhang (1989).

As an example of the generation of type 1 precedence relations, consider the example shown in figure 2, which has property 1. In this example, the infeasible states are

Ψ_X = {[false, true, false, false, true] , [true, false, false, true, false]}.

Therefore,

$$G(\boldsymbol{x}) = G(x_1, x_2, x_3, x_4, x_5) = \bar{x}_1 \cdot x_2 \cdot \bar{x}_3 \cdot \bar{x}_4 \cdot x_5 + x_1 \cdot x_2 \cdot x_3 \cdot x_4 \cdot x_5 \ .$$

The resulting precedence relations are

$$c_1 + c_3 + c_4 \rightarrow x_2 \cdot x_5 \ ,$$

$$c_2 + c_3 + c_5 \rightarrow x_1 \cdot x_4 \ .$$

A simpler set of precedence relations can be obtained if we also set nonstate vectors as don't-care conditions for the simplification. In this case, we obtain the precedence relations

$$c_1 \rightarrow x_2 \cdot x_5 \ ,$$

$$c_2 \rightarrow x_1 \cdot x_4 \ .$$

These sets of expressions are not unique; that is, other logically equivalent sets of precedence relations can be obtained as simplifications of

Precedence relations provide an implicit representation of assembly sequences . . .

this set. All these relations express our intuitive understanding of the problem that the stick must be connected prior to closing the receptacle and that the stick cannot be joined to the cap and handle without the receptacle already present.

Type 2 Precedence Relations: Connection-Connection

Type 2 precedence relations establish the occurrence of one connection prior to, or simultaneously with, the occurrence of another connection in the sequence. We use the notation $c_i < c_j$ to indicate that connection c_i must precede connection c_j, and we use the notation $c_i \leq c_j$ to indicate that connection c_i must precede or accompany the establishment of connection c_j. Furthermore, we use a compact notation for logical combinations of precedence relations; for example, we write $c_i < c_j \cdot c_k$ to mean $(c_i < c_j) \wedge (c_i < c_k)$, and we write $c_i + c_j < c_k$ to mean $(c_i < c_k) \vee (c_j < c_k)$. An assembly sequence whose representation as an ordered sequence of binary vectors is $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$ and whose representation as an ordered sequence of subsets of connections is $(\gamma_1\ \gamma_2 \ldots \gamma_{N-1})$ satisfies the precedence relationship $c_i < c_j$ if $c_i \in \gamma_a$, $c_j \in \gamma_b$, and $a < b$. Similarly, the sequence satisfies $c_i \leq c_j$ if $c_i \in \gamma_a$, $c_j \in \gamma_b$, and $a \leq b$. For example, for the assembly shown in figure 2, the assembly sequence whose representation as an ordered sequence of binary vectors is

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

and whose representation as an ordered sequence of subsets of connections is

$(\{c_1\}\ \{c_2, c_3\}\ \{c_4, c_5\})$

satisfies the precedence relationships $c_2 < c_4$ and $c_2 \leq c_3$ but does not satisfy the precedence relationships $c_2 < c_3$ and $c_2 \leq c_1$.

Each feasible assembly sequence of a given assembly can be uniquely characterized by a logical expression consisting of the conjunction of precedence relationships between the establishment of one connection and the establishment of another. Given an assembly made of N parts whose graph of connections is $\langle P, C \rangle$, let

$$\{(\gamma_{11}\ \gamma_{21} \ldots \gamma_{(N-1)1}), (\gamma_{12}\ \gamma_{22} \ldots \gamma_{(N-1)2}), \ldots, (\gamma_{1M}\ \gamma_{2M} \ldots \gamma_{(N-1)M})\}$$

be a set of M ordered sequences of subsets of connections that represent feasible assembly sequences. Then a correct and complete logical expression representing these sequences is

$$\prod_{i=1}^{L} \sum_{j=1}^{M} [H_{ij} \leq c_i \leq T_{ij}] ,$$

where

$$H_{ij} = \prod_{k=1}^{L} \lambda_{ik}, \text{ with } \lambda_{ik} = \begin{cases} c_k \text{ if } c_k \in \gamma_{lj} \text{ and } l \leq i \\ \text{true otherwise} \end{cases}$$

$$T_{ij} = \prod_{k=1}^{L} \lambda_{ik}, \text{ with } \lambda_{ik} = \begin{cases} c_k \text{ if } c_k \in \gamma_{lj} \text{ and } l \geq i \\ \text{true otherwise} \end{cases} .$$

Σ and Π again represent logical OR and AND, respectively.

Although it is often possible to directly reduce this logical expression, there is another interesting simplification that arises using property 1 and the assumption that all states are feasible. Under these assumptions, the following simplified set of type 2 precedence relations holds:

$$\prod_{i=1}^{L} \left[c_i \leq \sum_{j=1}^{M} T_{ij} \right] \left[\sum_{j=1}^{M} H_{ij} \leq c_i \right] .$$

This simplified set of precedence relations can also be proven for a more general set of assumptions leading to an independence property 2. Property 2 will not be discussed here because of space considerations.

We can illustrate this simplified set of precedence relations as follows. The assembly shown in figure 2 has both property 1 and feasible states. Based on the set of assembly sequences shown in figure 3, we can write type 2 precedence relations of the following form:

$$c_1 \leq c_2c_3c_4c_5 + c_2c_3c_4c_5 + c_3c_4c_5 + c_3c_5 + c_2c_4c_5 + c_2 + c_3c_5 + c_2 + c_2c_3c_4 + c_2$$

and

$$true + true + c_2c_3 + c_2c_3c_4c_5 + c_2c_3 + c_2c_3c_4c_5 + c_2c_3c_4c_5 + c_2c_3c_4c_5 + c_5 + c_2c_3c_4c_5 \leq c_1 .$$

By writing these relations for all $i = 1, \ldots, 5$, the resulting set of expressions can be simplified to obtain one nonredundant relation:

$$c_3 \leq c_1c_5 + c_2c_4 .$$

This type 2 precedence relation also specifies correctly and completely the set of feasible assembly sequences for the example in figure 2. It corresponds to our intuitive interpretation of the precedence that requires that the connection between stick and receptacle be established before the ends are connected.

Precedence relations provide an implicit representation of assembly sequences in the sense that decisions regarding the next step in the sequence can be made through local consideration of the current and previous states. However, an assembly sequence itself is an explicit representation of the assembly sequence, and if a new state is entered, an

entire sequence must be generated to guarantee its correctness. This distinction becomes important in the implementation of assembly sequence plans in real-time execution. In this mode of operation, the choice of a next-correct operation can be made without recourse to explicit planning of the entire sequence. However, using the types of precedence relations we have defined, there is no guarantee that a next-correct step is also going to be a step leading to a complete feasible sequence. Therefore, we have defined a real-time property of assembly sequence representations that requires that all feasible assembly subsequences that start at the initial state have a non-empty set of following subsequences that reach the goal. Zhang (1989) describes algorithms that generate precedence relations that guarantee the real-time property to hold.

An example of an assembly in which the type 1 precedence relations do not provide the real-time property is shown in figure 7. In this case, the extended algorithms are required to generate a set of precedence relations guaranteeing the real-time property.

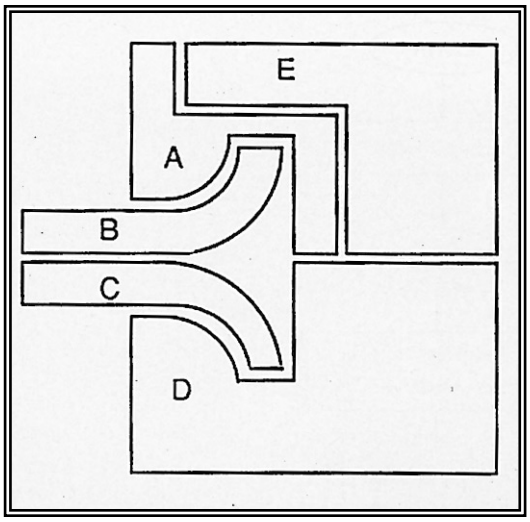


Figure 7. Example of an Assembly That Does Not Have the Real-Time Property for Type 1 Precedence Relations.

Although B and C can be individually inserted at any time, if B-C are connected early in the sequence, several additional feasible steps can be taken but do not lead to a feasible sequence. A deadend is reached when B-C tries to mate with A-E-D. The real-time precedence relations for this example are generated by our extended algorithm (Zhang 1989).

Evaluation and Selection of Assembly Plans

The assembly planning algorithms described in previous sections generate the set of all feasible assembly sequences. Formally defining these structures and algorithms that incorporate feasibility predicates is a necessary step toward developing techniques that can be used to evaluate and select plans for particular applications. In this section, we describe several approaches to the evaluation and selection of plans.

In practice, we would like to select a set of candidate assembly plans that most nearly meet our needs for a particular purpose. This selection requires the definition of evaluation measures and the implementation of search techniques to select appropriate plans. This evaluation and selection process would ideally occur in parallel with the generation of the plans themselves. The combinatorial explosion in the number of possible sequences makes it desirable to limit the search as early as possible and, therefore, to incorporate these evaluation measures into the search as early as possible. In this section, we summarize results for three different evaluation functions.

First, we would often like to choose assembly sequences that both minimize the complexity of the task execution as well as the complexity of the fixturing and manipulation to maintain the intermediate subassembly states. One possibility for such an evaluation function is a weighted combination of the complexity of the assembly tasks and the relative degrees of freedom of the parts in the intermediate subassemblies:

$W(t) = k_D \cdot D(n) + k_C \cdot C(h) + W(t_1) + W(t_2)$, where

$D(n)$ = the measure of the relative degrees of freedom of the parts of the subassembly, and n is the number of degrees of freedom.

$C(h)$ = the measure of the complexity of the assembly task whose corresponding AND-arc is h.

k_D = the weight given to the relative degrees of freedom of the subassembly parts.

k_C = the weight given to the task's complexity measure.

t_1 and t_2 are subtrees.

Subassemblies in which the number of relative degrees of freedom of the parts is high are more difficult to manipulate because there are fewer orientations in which they are stable, and there are fewer options for grasping. A variety of factors can be included in a measure of complexity of assembly tasks: time duration, reliability, fixture requirements, and cost of resources. As one example, we established a ranking of assembly tasks based on threaded contacts, cylindrical contacts, and planar contacts. Either exhaustive or heuristic

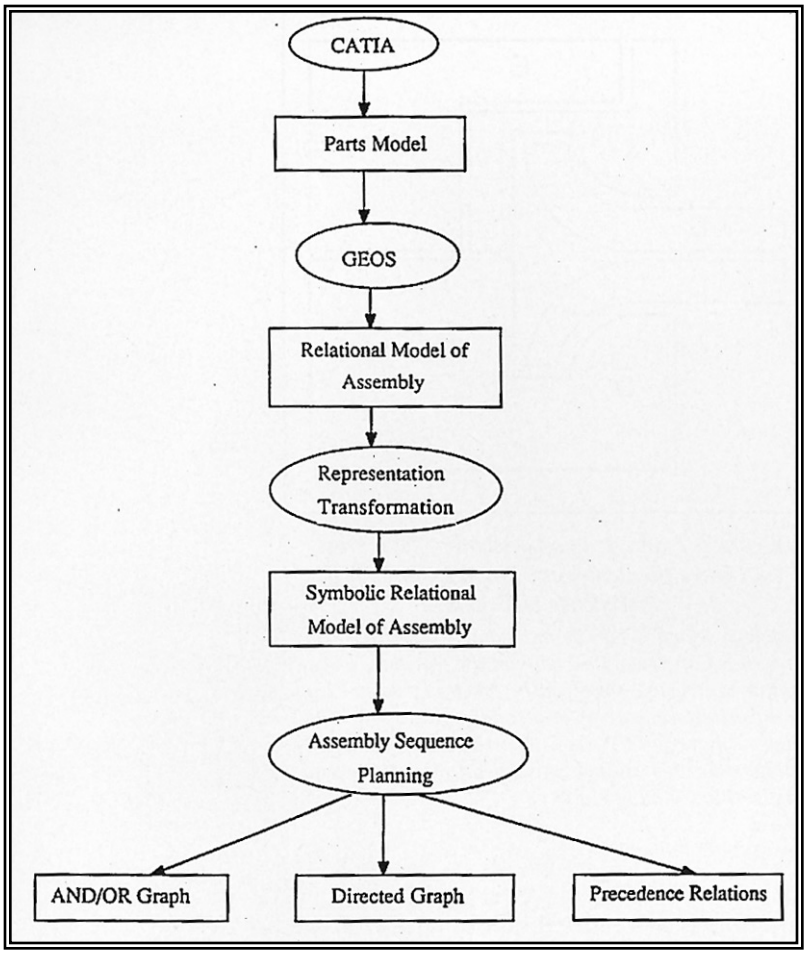


Figure 8. Overview of the Pleiades Environment for Assembly Sequence Planning.

TREE	COST
A-B-C	11
B-A-C	13
B-C-A	13
C-B-A	11
D-A-C	15
D-C-A	14
C-D-A	14
E-C-A	15
E-A-C	14
A-E-C	14

Table 1. Tables of Costs Using the Weighted Evaluation Function for the Example in Figure 2.
The preferred sequences were those that attached the cap or handle to the receptacle first, providing a stable subassembly for insertion of the stick.

search techniques can be used to explore the alternative sequences relative to these evaluation functions. In this case, an evaluation function *W* can be defined as an admissible heuristic function and incorporated into a heuristic search technique such as the AO* algorithm. Based on these search techniques, the resulting ranking of alternative sequences in terms of these evaluation functions is shown in table 1.

A second possible metric to assess the quality of an assembly sequence is the number of distinct sequences in which the assembly tasks can be executed. For some applications, it might be desirable to have one fixed set of assembly tasks, all of which are executed. Instead of allowing all possibilities given by the AND/OR graph, it might be preferable to allow only the possibilities given by one assembly tree. The assembly tree that allows the maximum number of distinct sequences is preferred because it gives more flexibility to task scheduling. Given an assembly tree, the number of distinct sequences in which the assembly task can be executed can be recursively computed. For this evaluation function, an admissible heuristic can also be found. The resulting evaluation of alternative assembly trees from figure 2 suggests that sequences D-C-A/C-D-A and E-A-C/A-E-C from figure 3 are each represented by the same solution tree for the assembly task, and the other six assembly trees allow only one sequence.

A third possibility for a metric to assess the quality of an assembly tree is its depth. Assuming that the assembly work station operates in cycles during which one or more assembly tasks is executed, the depth of an assembly tree is given by the minimum number of cycles that are required to complete the assembly. Given an assembly tree, the depth metric yields an admissible heuristic function that can be implemented as an efficient search technique. For the example in figure 2, the assembly trees found using metric 2 also have depth two because it is possible to simultaneously execute two assembly tasks and, therefore, to complete the assembly in two cycles. Of course, simultaneity of assembly tasks requires the availability of resources, and this metric would only be used if the required resources are available.

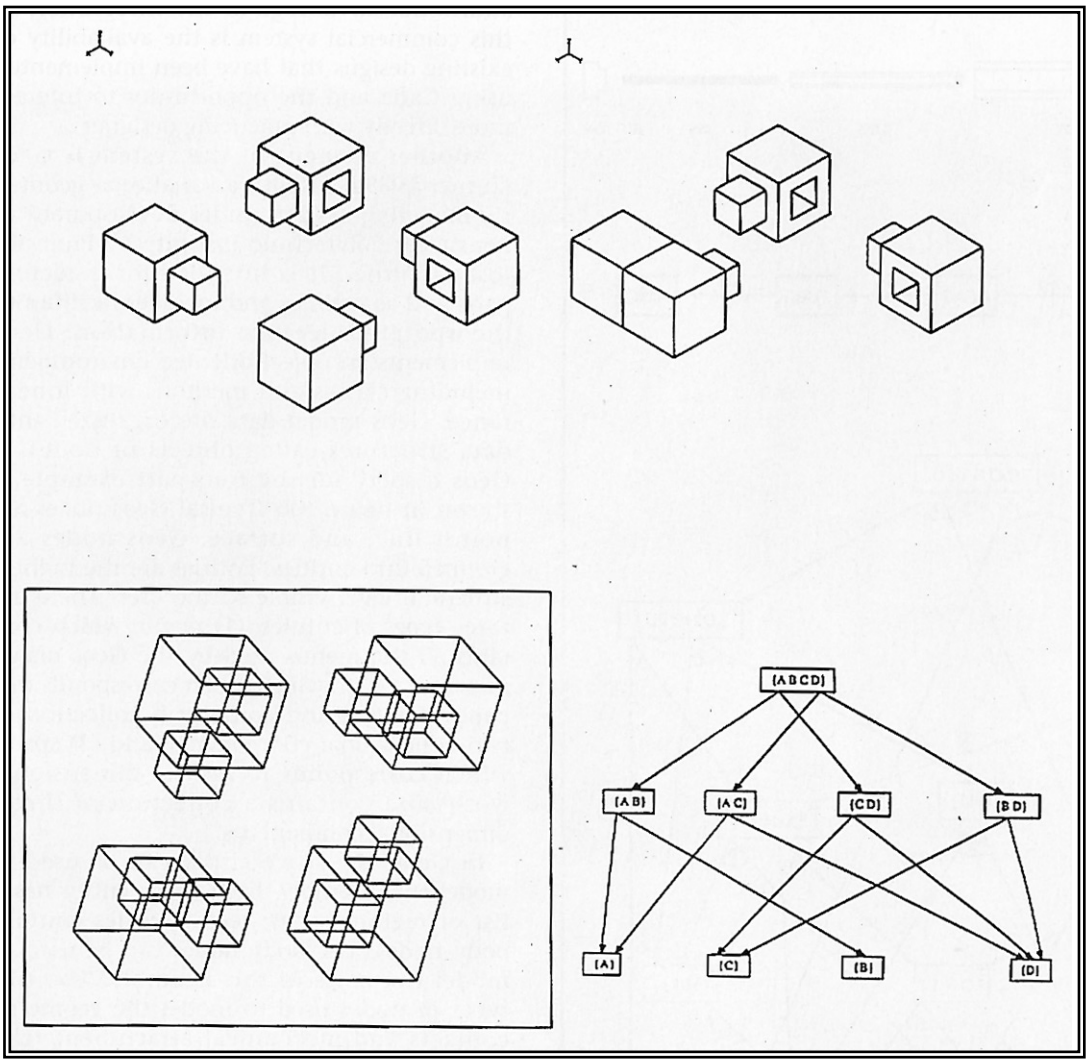


Figure 9. Example of the Sequence of Procedures in the Pleiades System.

Figure 9a shows the output of the Catia system showing the parts solid models, Figure 9b shows the Geos display of the object-oriented model of assembly relations, and Figure 9c shows the AND/OR graph output of the planning procedure.

. . . Pleiades has been used to demonstrate initial experiments in transferring a CAD-based parts description to a fully developed assembly plan.

The Pleiades System

Although a fully automated assembly system is one goal of the techniques described here, many of these approaches will be used first in an interactive mode, where the user is a critic for alternative assembly sequences generated automatically from CAD-based descriptions. The first generation of such an interactive assembly planning environment integrates a set of software modules for parts modeling and planning. The system, which we call Pleiades, has been used to demonstrate initial experiments in transferring a CAD-based parts description to a fully developed assembly plan.

Figure 8 illustrates the elements of the Pleiades system. Catia, a solid modeling and mechanical CAD design system available commercially from IBM, is the first element. Catia uses both constructive solid geometry and boundary representations and provides facilities for Boolean operations on solid parts models. Several parts can be built in the same model and viewed simultaneously, as shown in figure 9. However, there is no explicit representation for assembly relationships in Catia, and relative parts positions must be described to represent assembly relationships. The partially assembled parts for the same example are also shown in figure 9. For our purposes, Catia provides a convenient tool for

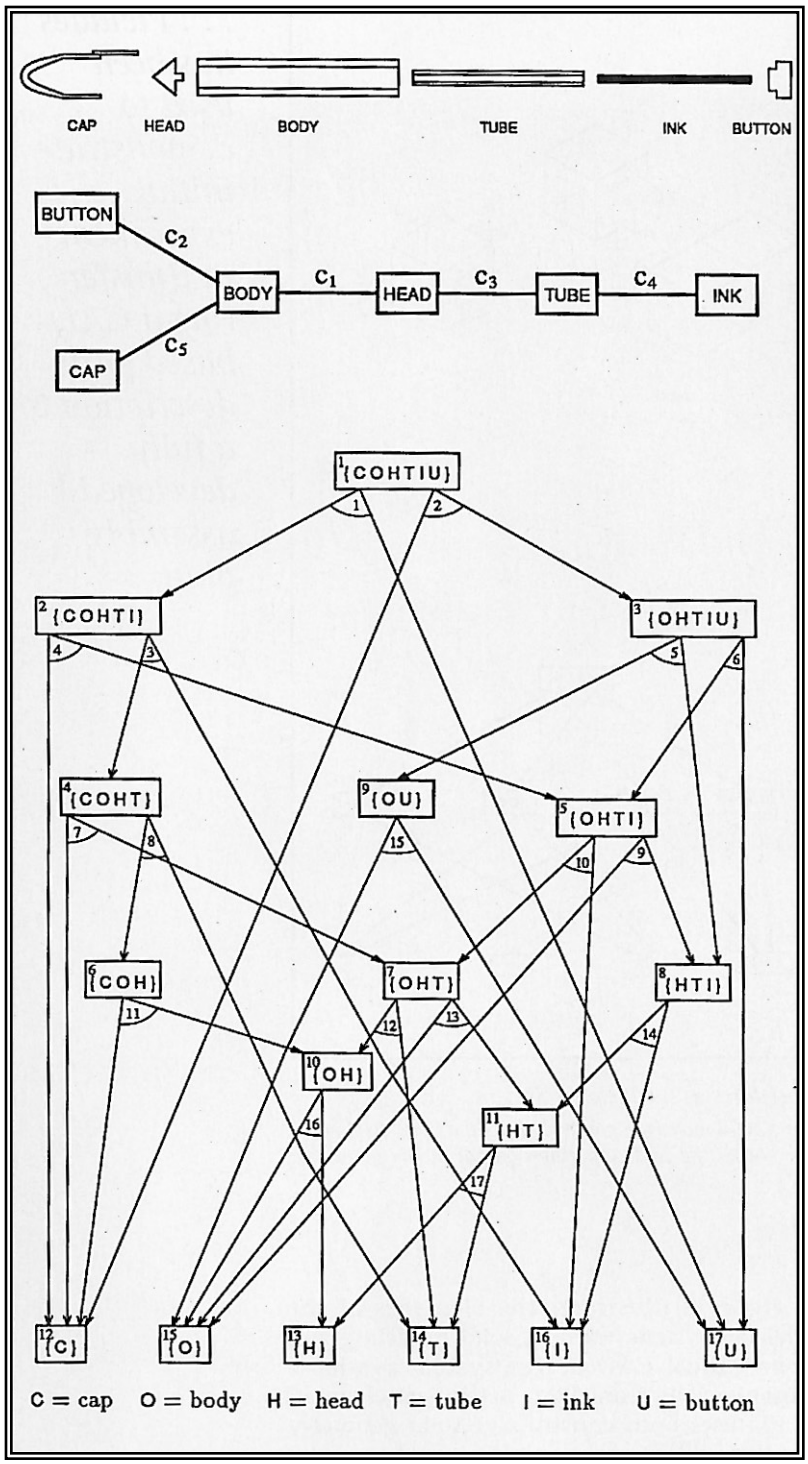


Figure 10. Ball-Point Pen Example.

generating and manipulating solid parts models. It provides sufficient boundary model information to generate many assembly relationships and provides good capabilities for visualization of parts and subassemblies. An additional advantage of the integration of this commercial system is the availability of existing designs that have been implemented using Catia and the opportunity to interact more directly with practicing designers.

Another element of the system is Geos (Turner 1989a, 1989b), a variational geometric modeling system under development at Rensselaer Polytechnic Institute by Professor Joshua Turner. It is intended for modeling parts and assemblies and includes facilities to incorporate tolerance information. Geos implements an object-oriented environment, including classes and methods with inheritance. Geos model data are organized into data structures called objects or nodes. A Geos display for the four-part example is shown in figure 10b. Typical Geos nodes are point, line, and surface. Geos nodes are grouped into entities. Entities are the highest structural level visible to the user. There are three types of entities: (1) menu, which contains all the menus used by the Geos dialog manager; (2) drawing, which corresponds to a paper drawing and contains a collection of two-dimensional components; and (3) space, which corresponds to a three-dimensional world and contains a collection of three-dimensional components.

In Geos, the space entity can be used to model the assembly. Each space entity has a list of region nodes; region nodes contain body nodes. The body nodes can be used to model the parts in the assembly. The two types of nodes used to model the geometric contacts and mechanical attachment relationships between the parts are those defined previously in our relational model (see Assembly Sequence Planning). A contact node models the contacts from the relational model and stores the pointers to the two body nodes and the pointers to the contacting surfaces. An attachment node includes the pointer to the agent nodes and the target nodes. In the current mode of operation, Geos imports parts models from Catia and converts them directly to Geos representation. The assembly relationships are input interactively.

A third element of the Pleiades system is the relational model. This model, which is used in the assembly planning software modules, is obtained by data format conversion from the Geos C environment to a Lisp environment. The resulting symbolic relational model contains all the relational and attribute information needed for assembly sequence planning, including evaluation of feasibility predicates for local geometric feasibility and attachments.

. . . Geos . . . a variational geometric modeling system . . . is intended for modeling parts and assemblies and includes facilities to incorporate tolerance information.

The assembly sequence planner uses the symbolic representation of the relational model to generate the AND/OR graph representation of all feasible assembly plans based on the evaluation of local geometric and attachment feasibility predicates. The AND/OR graph for the four-part example is shown in figure 9c. The explicit assembly sequence lists and the directed graph of assembly states can be generated from this representation.

Type 1 and type 2 precedence relations in the system are generated from the AND/OR graph representation using implementations of the algorithms described in Zhang (1989). For the four-part example in figure 9, these are, respectively,

$$c_1 \rightarrow x_3 \cdot x_4 \qquad c_2 \rightarrow x_1 \cdot x_3 \qquad c_3 \rightarrow x_2 \cdot x_4$$
$$c_4 \rightarrow x_1 \cdot x_2$$

and

$$c_1 \leq c_2 \cdot c_3 + c_4 \qquad c_2 \leq c_1 \cdot c_4 + c_3$$
$$c_3 \leq c_1 \cdot c_4 + c_2 \qquad c_4 \leq c_2 \cdot c_3 + c_1 \ .$$

Another example of the application of these planning tools is the ball-point pen shown in figure 10. This example was used by both Bourjault (1984) and DeFazio and Whitney (1988) in their work on interactive sequence planning. This example has six parts and five connections, with a resulting 12 feasible sequences for assembly. The AND/OR graph has 17 nodes, and the directed graph has 24 nodes. The type 1 and type 2 precedence relations that are generated for this example are, respectively,

$$c_4 \rightarrow x_1 \cdot x_2 \qquad c_3 \rightarrow x_4 \qquad c_1 \rightarrow x_5$$

and

$$c_1 \leq c_5 \qquad c_3 \leq c_4 \qquad c_4 \leq c_2 + c_1 \ ,$$

where the type 1 precedence relations are identical to those obtained from the interactive methods in DeFazio and Whitney (1988).

.

Conclusions

This article summarizes work in progress on the development of a theoretical framework and implementation of assembly sequence planning tools. The principal focus has been the formalization of assembly sequence plan representations, the proof of equivalence of different representations, and the implementation of algorithms to generate and transform these plan representations. These representation tools and algorithms form the basis for implementation in a system of software modules that provides an environment for interactive design and evaluation of assembly system design alternatives.

A number of additions and extensions to the current system are of both theoretical and practical importance. First are feasibility predicates. Within the framework we described, a large number of different resource-independent and resource-dependent feasibility predicates can be defined. For practical application, we need to incorporate a global geometric feasibility algorithm. There are several candidates here, but many existing algorithms that plan explicit paths are too computationally complex. An extended state stability criterion is also an important extension but will require additional research.

Second is hierarchy. Many assembly products are designed with natural hierarchies of subassemblies to maintain the stability of subunits, resulting in both functional and manufacturing advantages. Such hierarchies fit naturally into the AND/OR graph framework and can be used to simplify the search procedure by enforcing subgoals.

Third are uncertainty and tolerances. Assembly product designs have specified tolerances on the dimensions and shapes of parts. These tolerances are usually matched to the functional specification of the parts and often reflect an implicit assumption by the designer about assembly sequencing. Incorporation of tolerance information into feasibility predicates for sequence planning is an important extension to our current work. We expect this analysis to lead to formal methods to incorporate other sources of uncertainty resulting from fixtures or manipulators.

Fourth are mechanisms. In our current

A number of additions and extensions to the current system are of both theoretical and practical importance.

analysis and experiments, we do not permit an assembly to occur in more than one final state. In practice, many assemblies are complex mechanical mechanisms whose function depends on the occurrence of multiple states, and the assembly sequence can depend on the ability to alter state during the assembly process. These considerations fall within the framework that we defined, but we have not attempted to implement these considerations.

Fifth is efficiency. The emphasis in this work has been in complete and correct representations of all feasible assembly sequences. In practice, the combinatorial search of all possible sequences is often prohibitive. We view this complete and correct specification as a necessary precursor to more efficient decomposition and search techniques that are implemented in a partial representation space. Our experiments with evaluation criteria, discussed in Evaluation and Selection of Assembly Plans, provide examples of heuristic search within the AND/OR graph space without requiring that the entire graph be generated.

Sixth is manipulation planning. The implementation of an assembly system requires links between the sequence planner and the manipulation planner or task-level programmer. Natural extensions of the structure described here incorporate feasibility tests on resource-dependent properties such as gripper geometry or force exerted. The detailed planning of manipulation and fine-motion actions is outside the direct scope of this work but will be essential for the automation of such systems. Although each of these topics is a difficult research topic in itself, the abstracted representation of manipulation capability must be present for adequate planning of sequence based on resource-dependent considerations.

Seventh is parts design. We emphasized the planning of feasible sequences given a specification of parts and assembly relationships. Clearly, parts redesign is one of the important links to successful manufacturing systems. A Pleiades-type planning environment would permit the exploration of parts geometry and assembly modifications using the Catia system and an opportunity to evaluate the impact of design changes on assembly plans.

Acknowledgments

This work has been supported by the New York State Center for Advanced Technology in Automation and Robotics and the NASA Center for Intelligent Robotics Systems for Space Exploration at Rensselaer Polytechnic Institute. Additional support has been received from the Conselho Nacional de Desenvolvimento Cientifico e Tecnologico (Brazil), The Jet Propulsion Laboratory of the California Institute of Technology, and The Robotics Institute of Carnegie-Mellon University.

The authors would like to express their appreciation to Joshua Turner at Rensselaer Polytechnic Institute for his helpful discussions and suggestions regarding the integration of the Catia and Geos systems.

References

Bonenschanscher, N. 1988. Subassembly Stability. In Proceedings of the Seventh National Conference on Artificial Intelligence, 780–785. Menlo Park, Calif.: American Association for Artificial Intelligence.

Bourjault, A. 1984. Contribution a une Approche Methodologique de L'Assemblage Automatise: Elaboration Automatique des Sequences Operatoires ("Contribution to the Methodology of Automated Assembly: Automatic Generation of Operations Sequences"), These d'Etat, Universite de Franche-Comte.

Chapman, D. 1987. Planning for Conjunctive Goals. *Artificial Intelligence* 32:333–377.

DeFazio, T. L., and Whitney, D. E. 1988. Simplified Generation of All Mechanical Assembly Sequences. *IEEE Journal of Robotics and Automation* RA-3(6): 640–658. Also 1988. Corrections. *IEEE Journal of Robotics and Automation* RA-4(6): 705–708.

Eastman, C. M. 1981. *The Design of Assemblies.* SAE Technical Paper Series. Detroit, Mich.: Society of Automotive Engineers.

Fikes, R., and Nilsson, N. 1972. STRIPS: A New Application of Theorem Proving to Problem Solving. *Artificial Intelligence* 2:189–208.

Homem de Mello, L. S. 1989. Task Sequence Planning for Robotic Assembly. Ph.D. diss., Electrical and Computer Engineering Department, Carnegie-Mellon University.

Homem de Mello, L. S., and Sanderson, A. C. 1990. A Correct and Complete Algorithm for the Generation of Mechanical Assembly Sequences. *IEEE Transactions on Robotics and Automation*. Forthcoming.

Homem de Mello, L. S., and Sanderson, A. C. 1989a. Precedence Relationship Representations of Mechanical Assembly Sequences. In Proceedings of the Second NASA Workshop on Space Telerobotics, 129–138. Pasadena, Calif.: Jet Propulsion Laboratory, California Institute of Technology.

Homem de Mello, L. S., and Sanderson, A. C. 1989b. Representations of Assembly Sequences. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence. Menlo Park, Calif. International Joint Conferences on Artificial Intelligence.

Homem de Mello, L. S., and Sanderson, A. C. 1988. Automatic Generation of Mechanical Assembly Sequences, Technical Report, CMU-RI-TR-88-19, The Robotics Inst., Carnegie-Mellon Univ.

Homem de Mello, L. S., and Sanderson, A. C. 1987. Task Planning and Control Synthesis for Flexible Assembly Systems. In *Machine Intelligence and Knowledge Engineering for Robotic Applications*, NATO ASI Series, Vol. F33, eds. A. K. C. Wong and A. Pugh, 331–353. Berlin: Springer-Verlag.

Homem de Mello, L. S., and Sanderson, A. C. 1986. AND/OR Graph Representation of Assembly Plans. In Proceedings of the Fifth National Conference on Artificial Intelligence, 780–785. Menlo Park, Calif.: American Association for Artificial Intelligence.

Korf, R. E. 1987. Planning as Search: A Quantitative Approach. *Artificial Intelligence* 33:65–88.

Lee, K., and Gossard, D. C. 1985. A Hierarchical Data Structure for Representing Assemblies: Part 1. *CAD* 17(1): 15–19.

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

Lozano-Perez, T., and Wesley, M. 1979. An Algorithm for Planning Collision-Free Paths among Polyhedral Obstacles. *Communications of the ACM* 22:560–570.

Lui, M. M. 1988. Generation and Evaluation of Mechanical Assembly Sequences Using the Liaison-Sequence Method. Master's thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology.

Sacerdoti, E. D. 1973. Planning in a Hierarchy of Abstraction Spaces. In Proceedings of the Third International Joint Conference on Artificial Intelligence, 412–422. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Sanderson, A. C., and Homem de Mello, L. S. 1990. Automatic Generation of Mechanical Assembly Sequences. In *Geometric Modeling for Product Engineering*, eds. M. Wozny, J. Turner, and K. Preiss, 461–482. New York: Elsevier.

Turner, J. 1989a. GEOS Design Notes, Rensselaer Design Research Center, Rensselaer Polytechnic Institute.

Turner, J. 1989b. GEOS User Notes, Rensselaer Design Research Center, Rensselaer Polytechnic Institute.

Wilkins, D. 1984. Domain-Independent Planning: Representation and Plan Generation. *Artificial Intelligence* 22:269–301.

Zhang, H. 1989. Generation of Precedence Relations for Mechanical Assemblies, Master's thesis, Rensselaer Polytechnic Institute.

Arthur C. Sanderson is professor and chairman of the Electrical, Computer, and Systems Engineering Department at Rensselaer Polytechnic Institute, Troy, NY 12180. He is the author of over 130 technical publications and proceedings. His current research interests include planning systems for robotics and automation, sensor-based control, computer vision, and applications of knowledge-based systems. He is currently president of the Institute of Electrical and Electronic Engineers (IEEE) Robotics and Automation Society and was general chairman of the 1989 IEEE International Symposium on Intelligent Control.

Luiz S. Homem de Mello is at the Jet Propulsion Laboratory of the California Institute of Technology, working in the Robotic Intelligence Group. He received his Ph.D. in electrical and computer engineering from Carnegie-Mellon University, where he did research on automatic planning for robotic assembly. In addition to assembly planning, his research interests include robotics, intelligent control, and AI. He is a member of the American Association for Artificial Intelligence and the Institute of Electrical and Electronic Engineers.

Hui Zhang is a graduate student in the Department of Electrical Engineering at the University of California at Berkeley. He received his B.S. in computer science from Beijing University and his M.S. in computer engineering from Rensselaer Polytechnic Institute.