REPRESENTATIONS OF ASSEMBLY SEQUENCES

L. S. Hornem de Mello
The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213-3890

A. C. Sanderson
Electrical, Computer, and Systems Engineering Department
Rensselaer Polytechnic Institute
Troy, New York 12180-3590

Abstract

This paper analyses four representations for assembly sequences which are based on directed graphs, on AND/OR graphs, on establishment conditions, and on precedence relationships. The latter includes two types: precedence relationships between the establishment of one connection between parts and the establishment of another connection, and precedence relatioships between the establishment of one connection and states of the assembly process. The paper discusses how each representation is related to the others. The correctness and completeness of these representations are also addressed. The results presented are needed to prove the correctness and completeness of algorithms for the generation of mechanical assembly sequences.

1. Introduction

Choosing the representation of assembly sequences is an important decision both in creating an assembly sequence planner and in designing an intelligent control for the assembly process.

Several methodologies for representing assembly sequences have been utilized. These include representations based on directed graphs, on AND/OR graphs, on establishment conditions, and on precedence relationships. Those based on directed graphs and on AND/OR graphs are explicit representations since there is a mapping from the assembly tasks into the elements of the representations. Those based on establishment conditions and on precedence relationships are implicit representations because they consist of conditions that must be satisfied by the assembly sequences. Despite the diversity of representations, there is a lack of understanding of how each representation maps into the others and of how one representation can be derived from the others. Furthermore, the correctness and completeness of the implicit representations have not received adequate attention. By correctness of the representation we mean that only feasible sequences satisfy' the conditions. By completeness we mean that all the feasible sequences satisfy the conditions. A prerequisite for a proof of correctness and completeness of algorithms that generate assembly sequences [Bourjault 84, De Fazio and Whitney 87, Homem de Mello and Sanderson 89] is a proof that the representation of assembly sequences used is correct and complete.

This paper analyses these four representations and shows how they are interrelated. The correctness and completeness of these representations are also addressed.

2. Background

A mechanical assembly is a composition of parts interconnected forming a stable uniL Each part is a solid object. Parts are interconnected whenever they have one or more surfaces in contact. Surface contacts between parts reduce the degrees of freedom for relative motion. A cylindrical contact, for example, prevents any relative motion that is not a translation along the axis or a rotation around the axis. Attachments may act on surface contacts and eliminate all degrees of freedom for relative motion. For example, if a cylindrical contact has a pressure-fit attachment, then no relative motion between the parts is possible.

A subassembly is a nonempty subset of parts that either has only one element or is such that every part has at least one surface contact with another part in the subset Although there are cases in which it is possible to join the same pair of parts in more than one way, a unique assembly geometry will be assumed for each pair of parts. This geometry corresponds to their relative location in the whole assembly. A subassembly is said to be stable if its parts maintain their relative position and do not break contact spontaneously. All one-part subassemblies are stable.

The assembly process consists of a succession of tasks, each of which consists of joining subassemblies to form a larger subassembly. The process starts with all parts separated and ends with all parts properly joined to form the whole assembly. For the current analysis, it is assumed that exactly two subassemblies are joined at each assembly task, and that after parts have been put together, they remain together.

It is also assumed that whenever two parts are joined all contacts between them are established. Due to this assumption, an assembly can be represented by a simple undirected graph (P, C) in which $P = \{p_1, p_2, \ldots, P_N\}$ is the set of nodes, and $C = \{c_1, c_2, \ldots, c_L\}$ *is the set of edges.* Each node in *P* corresponds to a part in the assembly, and there is one edge in *C* connecting every pair of nodes whose corresponding parts have at least one surface contact. The elements of *C* are referred to as *connections,* and the graph *{P*, C) is referred to as *graph of connections.* Figure 1 shows an assembly in exploded view, and figure 2 shows its corresponding graph of connections. The state of the assembly process can be characterized by the connections that have already been established, and it can be represented by an L-dimensional binary vector $x = [x_1, x_2, \ldots, x_L]$ in which the the component x_i is T or F respectively if the i^{th} connection is established in that state or not.

Furthermore, it is assumed that whenever a subassembly is formed all connections between its parts are established. Therefore, any subassembly can be characterized by its set of parts, and any state of the assembly process can be characterized by a partition of the set of parts of the whole assembly. For example, the initial state of the assembly process of the assembly shown in figure 1 is characterized by | |CAP), {RECEPTACLE), {STICK}, {HANDLE)) whereas the final state is characterized by { {CAP, RECEPTACLE, STICK, HANDLE)). In the rest of this paper, references to subsets of parts should be understood as references to the subassemblies made up of those parts. It will always be clear from context what the whole assembly is.

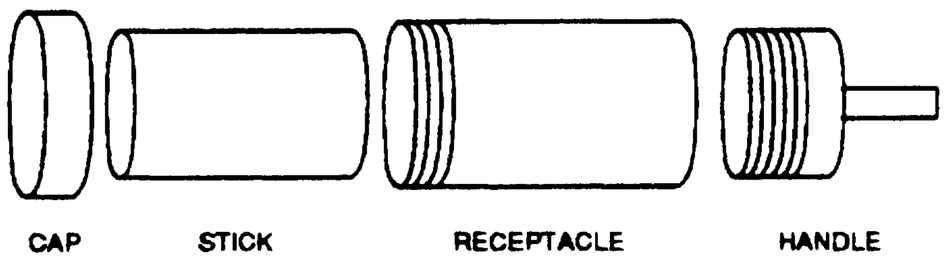


Figure 1: A simple product in exploded view.

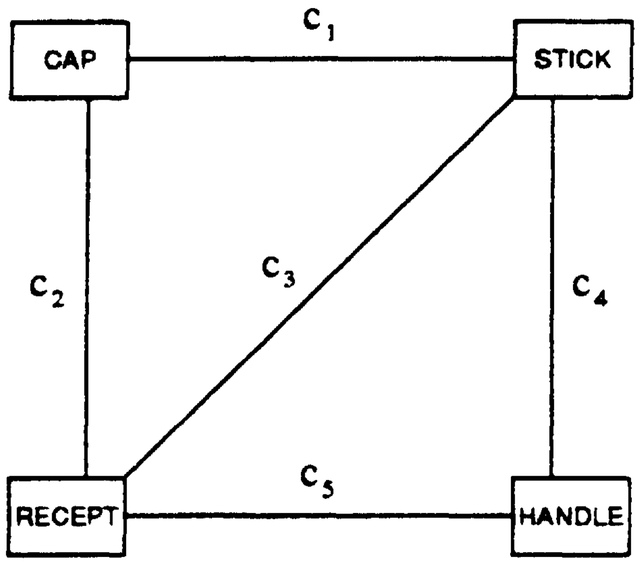


Figure 2: The graph of connections for the product shown in Figure 1

There are partitions of the set of parts of the whole assembly that cannot characterize a state of the assembly process. For example, the partition { {CAP, HANDLE}, {RECEPTACLE}, {HANDLE} } cannot characterize a state of the assembly process for the assembly shown in figure 1 because the subset {CAP, HANDLE} does not characterize a subassembly. Partitions that characterize a state of the assembly process will be referred to as *stale partitions,* and partitions that don't characterize a state will be referred to as *nonslate partitions.*

Similarly, not all L-dimensional binary vectors can characterize a state. For example, for the product shown in figure 1, the 5-dimensional binary vector [T, T, F, F, F] does not correspond to a state because if connections C_1 and C_2 are established then connection C_3 should also be established. L-dimensional binary vectors that characterize a state will be referred to as *state vectors* whereas L-dimensional binary vectors mat don't characterize a state will be referred to as *nonslate vectors.*

We will use the subassembly predicate *sa* to determine whether a subset of parts makes up a subassembly. The argument to this predicate is a subset of parts, and its value is either T or F depending on whether that subset of parts corresponds to a subassembly. For example, for the assembly shown in figure 1, *sa(* { RECEPTACLE. HANDLE }) = T, whereas *sa({* CAP, HANDLE }) = F. From the assembly's graph of connections it is straight forward to compute *sa* for any given subset of parts. Similarly, we will use the subassembly-stability predicate *st* to determine whether a subassembly described by its set of parts is stable. The computation of *st* has been addressed elsewhere [Blum et aL 70, Boneschanscher et al. 88].

Given two subassemblies characterized by their sets of parts 6 and θj, we say that joining θ_i and θ is an assembly task if the set $\theta_k = \theta_i U \theta_i$ characterizes a subassembly. For example, for the assembly shown in figure 1, if θ_i={ RECEPTACLE } and 0 = { HANDLE } then joining θ_i and θ jis an assembly task, whereas if θ_i={ CAP } and θ_j={ HANDLE) then joining 0. and 0 is not an assembly task. The subassemblies θ_i and θ_j are the *input* subassemblies of the assembly task, and θ_k is the *output* subassembly of the assembly task. Due to the assumption of unique geometry, an assembly task can be characterized by its input subassemblies only.

An assembly task is said to be *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 And an assembly task is said to be *mechanically* feasible if it is feasible to establish the attachments that act on the contacts between the two subassemblies. We will use the geometric-feasibility predicate *gf* and the mechanical-feasibility predicate *mf* to determine whether two subsets of parts characterize, respectively, a geometrically feasible and a mechanically feasible assembly task. These predicates take as argument a set of two subassemblies, each characterized by its set of parts. The computation of these predicates is discussed elsewhere [Homem de Mello and Sanderson 89].

Given an assembly that has *N* parts, an ordered set of N -1 assembly tasks 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 T_i is either a one-part subassembly or the output subassembly of a task that precedes T_i. To any assembly sequence T1,T2,.....,T_{N-1} there corresponds an ordered sequence S_1, S_2, • • • S_N *of* states of the assembly process. 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. And any two consecutive states s_i and s_{i+1} are such that only the two input subassemblies of task T_i are in s_i and not in s_{i+1}, and only the output subassembly of task T_i is in s_{i+1} and not in s_i. Therefore, an assembly sequence can also be characterized by an ordered sequence of states.

An assembly sequence is said to be feasible if all its assembly tasks are geometrically and mechanically feasible, and the input subassemblies of all tasks are stable.

Since one assembly sequence can be represented by an ordered list of tasks, it is possible to represent the set of all assembly sequences by a set of lists, each corresponding to a different assembly sequence. Since many assembly sequences share common subsequences, attempts have been made to create more compact representations that can encompass all assembly sequences. The next sections discuss different approaches towards representing all assembly sequences of a mechanical assembly.

3. Directed Graph Representation of Assembly Sequences

Given an assembly whose graph of connections is (P,C), a directed graph can be used to represent the set of all assembly sequences. The nodes in this directed graph correspond to stable state partitions of the set *P.* These are the partitions 0 of *P* such that if 0 € 6 then 0 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 partition corresponding to the second node. Also, there is only one subset θ_k in the state partition corresponding to the second node that is not in the state partition corresponding to the first node, and $\theta_k = \theta_i U \theta_j$Furthermore, the assembly task that joins θ_i and θ_j is feasible. Therefore, each edge corresponds to an assembly task. This graph is referred to as *directed graph of feasible assembly sequences,* and it can be formally defined as follows:

Definition 1: The directed graph of feasible assembly sequences of an assembly whose set of parts is *P* is the directed graph (*Xp*, *Tp*) in which

$$X_P = \{\Theta \mid [\Theta \in \Delta(P)] \wedge [\forall \theta(\theta \in \Theta) \Rightarrow sa(\theta) \wedge st(\theta)]\}$$

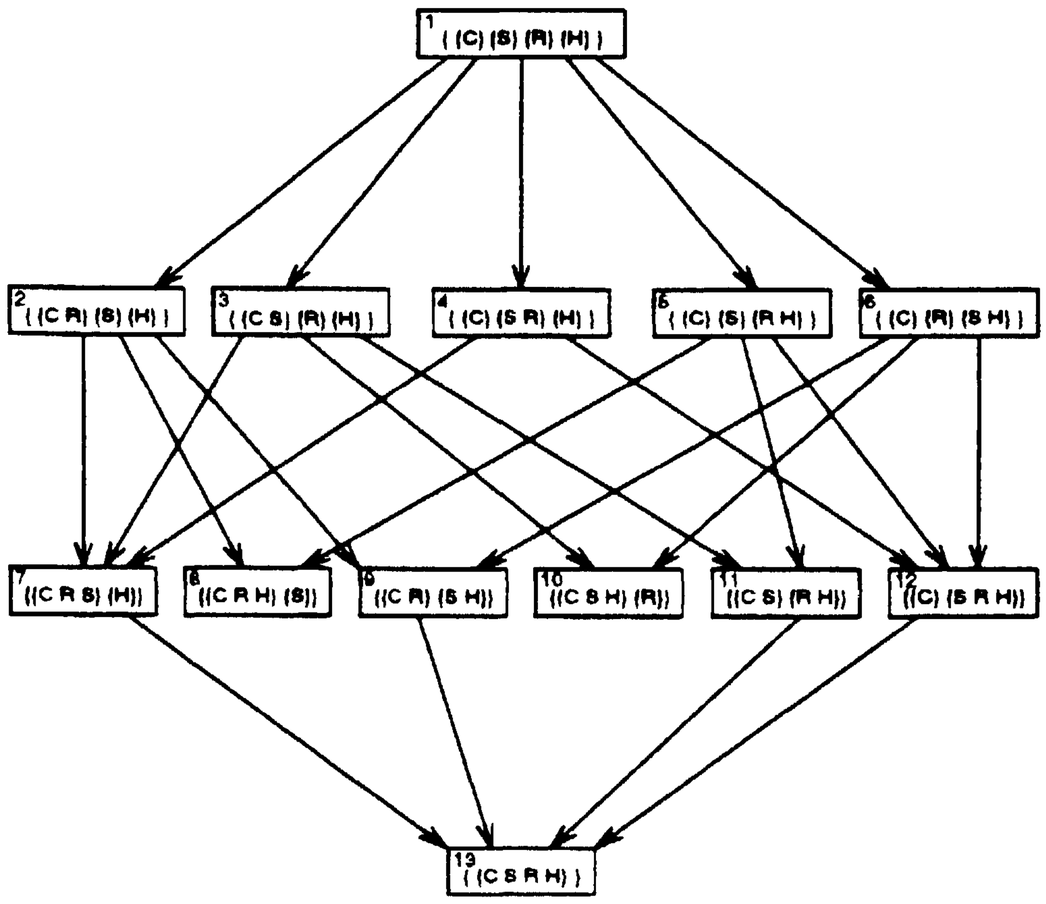
is the assembly's set of stable states, and

$$T_P = \{(\Theta_i, \Theta_j) \mid [(\Theta_i, \Theta_j) \in X_P \times X_P] \wedge [|\Theta_j - (\Theta_i \cap \Theta_j)| = 1] \wedge [|\Theta_i - (\Theta_i \cap \Theta_j)| = 2] \wedge [U(\Theta_i - (\Theta_i \cap \Theta_j)) \in \Theta_j - (\Theta_i \cap \Theta_j)] \wedge [mf(\Theta_i - (\Theta_i \cap \Theta_j))] \wedge [gf(\Theta_i - (\Theta_i \cap \Theta_j))]\}$$

is the assembly's set of feasible state transitions.

The notation A(P) has been used to represent the set of all partitions of P, and $U(\{A, B, \cdots, Z\})$=AUBU $\cdots$ UZ. As an example, figure 3 shows the directed graph of feasible assembly sequences for the product shown in figure 1.

A path in the directed graph of feasible assembly sequences (X_p, T_p) whose initial node is $\Theta_1 = \{\{p_1\}, \{p_2\}, \cdots, \{p_N\}\}$ and whose terminal node is $\Theta_F = \{\{p_1, p_2, \cdots, p_N\}\}$ corresponds to a feasible assembly sequence for the assembly P, and conversely. In such a path, the ordered sequence of edges corresponds to the ordered sequence of tasks, while the ordered sequence of nodes corresponds to the ordered sequence of states of the assembly process.



C = cap S = stick R = receptacle H = handle

Figure 3: Directed graph of feasible assembly sequences for the assembly shown in figure 1

4. AND/OR Graph Representation of Assembly Sequences

In our previous work [Homem de Mello and Sanderson 86], we introduced an AND/OR graph representation of assembly sequences. The nodes in this AND/OR graph are the subsets of *P* that characterize stable subassemblies. The hyperarcs correspond to the geometrically and mechanically 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 $\Theta_i \cup \Theta_j = \Theta_k$ and the assembly task characterized by θ_i and θ_j is feasible. This AND/OR graph can be formally defined as follows:

Definition 2: The AND/OR graph of feasible assembly sequences of an assembly whose set of parts is $P = \{P_1, p_2, \cdots, P_N\}$, is the AND/OR graph (Sp, Dp) in which

$S_p = \{\theta \varepsilon \quad Tl(P) \backslash sa(B) ASt(B)\}$

is the set of stable subassemblies, and

$$D_P = \{(\theta_k, \{\theta_i, \theta_j\}) \mid [\theta_i, \theta_j, \theta_k \in S_P] \wedge [mf(\{\theta_i, \theta_j\})] \wedge [gf(\{\theta_i, \theta_j\})] \wedge [U(\{\theta_i, \theta_j\}) = \theta_k]\}$$

is the set of feasible assembly tasks.

The notation $\prod (P)$ has been used to represent the set of all subsets of *P*. As an example, figure 4 shows part of the AND/OR graph for the assembly shown in figure 1. Each node in that graph is associated with a subset of parts that corresponds to a subassembly. There are only two stable subassemblies of the product shown in figure 1 whose corresponding nodes were not included in figure 4; they are the subassembly made up of the cap, the receptacle, and the handle, and the subassembly made up of the cap, the stick, and the handle. They were not included to avoid cluttering the figure.

From the AND/OR graph of feasible assembly sequences one can define feasible assembly trees as follows:

Definition 3: Given the AND/OR graph of feasible assembly sequences of an assembly whose set of parts is $P = \{p_1, p_2, \ldots\ldots p_N\}$, any AND/OR path having $\{p_1, p_2, \ldots\ldots P_N\}$ as its initial node, and having $\{p_1\}, \{p_2\}, \ldots\ldots (P_N)$ as its terminal nodes is a feasible assembly tree of that assembly.

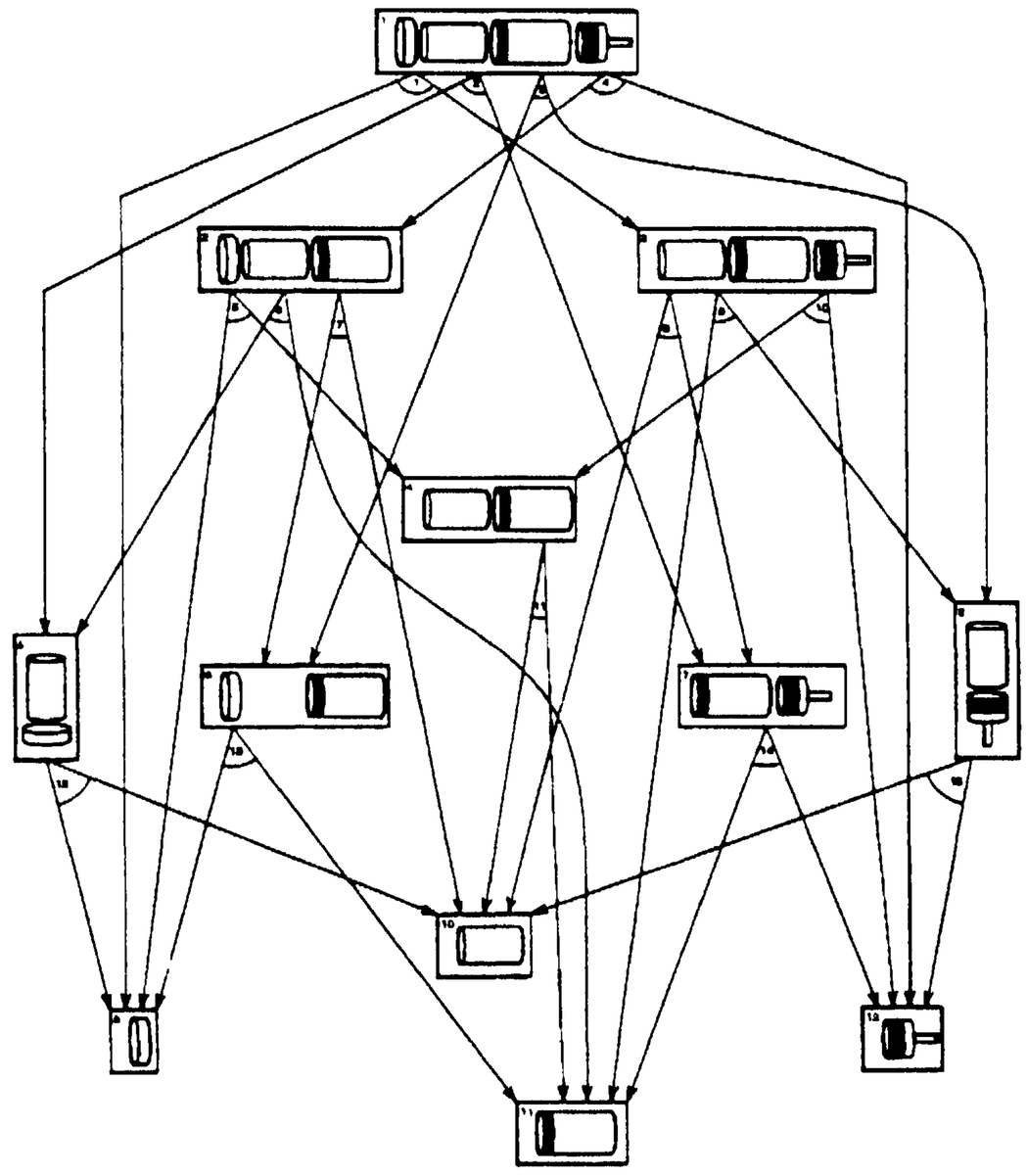


Figure 4: AND/OR graph of feasible assembly sequences for the assembly shown in figure 1.

An assembly tree induces a partial order among its hyperarcs: hyperarc $h \bullet$ is said to precede hyperarc h_j if there is a node n_k in the assembly tree such that *hi* is incident from n_k and h_i is incident to n_i. At least one sequence of the hyperarcs of an assembly tree is consistent with this partial order. Furthermore, every sequence of the hypearcs that is consistent with the partial order corresponds to a feasible assembly sequence.

- The Correspondence between the Directed Graph and the AND/OR Graph

Every feasible assembly sequence in the directed graph of feasible assembly sequences corresponds to a feasible assembly tree in the AND/OR graph of feasible assembly sequences. And every feasible assembly tree in the ANDADR graph of feasible assembly sequences corresponds to one or more feasible assembly sequences in the directed graph of feasible assembly sequences. The two theorems below establish the correspondence between assembly trees and assembly sequences. Proofs of these theorems are presented elsewhere [Homem de Mello 89].

Theorem 4: Given an assembly tree of an assembly, if h_1, h_2 .... , $h_i=(\sigma_i,,\Theta_i,)$, ... h_{N-1} is a sequence of all the hyperarcs of that assembly tree that is consistent with the partial order induced by the tree, then the sequence $\Omega_1,\Omega_2,\cdots,\Omega_N$, in which $\Omega_1 = \{ P_1), \{ P_2) \ldots\ldots \{P_2)\}$, and $\Omega_{i+1}=(\Omega_i-\Theta_i)\cup\{\sigma_i\}$ is a feasible assembly sequence of the assembly.

Theorem 5: If $\Omega_1,\Omega_2,\cdots,\Omega_N$ is an assembly sequence of an assembly whose set of parts is $P = \{p_1, p_2, \ldots\ldots p_N)$ and

$$S_P^a = \Omega_1 \cup \Omega_2 \cup \cdots \cup \Omega_N$$
$$\{\sigma_i\} = \Omega_{i+1} - (\Omega_i \cap \Omega_{i+1}) \text{ for } i=1,2,\cdots,N-1$$
$$\Theta_{\sigma_i} = \Omega_i - (\Omega_i \cap \Omega_{i+1}) \text{ for } i=1,2,\cdots,N-1$$
$$h_i=(\sigma_i,\Theta_{\sigma_i}) \text{ for } i=1,2,\cdots,N-1$$

$$H_P^a = \{h_1,h_2,\cdots h_{N-1}\}$$

then (S_p, H_p) is an assembly tree of that assembly.

The useful feature of the AND/OR graph representation for assembly sequences is that for assemblies of more than 5 parts it encompasses all possible assembly sequences with a reduced number of nodes than the directed graph of assembly states [Homem de Mello and Sanderson 88]. Furthermore, it explicitly shows the possibility of simultaneous execution of assembly tasks.

5. Establishment Condition Representation of Assembly Sequences

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 $E_i=\{x_1,x_2,\ldots\ldots x_{Ki}\}$ *be the set of states ftom*

which the i^{th} connection can be established without precluding the completion of the assembly. The *establishment condition* for the i^{th} connection is the logical function

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

where the sum and the product are the logical operations OR and AND respectively, and y_{kl} is either the symbol x_l if the l^{th} component of x_k is T, or the symbol x_l if the l^{th} component of x_k is F. Clearly, every element x_k of E_i is such that $F_i(x_k) = T$. It is often possible to simplify the expression of F_i using the rules of boolean algebra.

The establishment conditions can be obtained from the directed graph of feasible assembly sequences by systematically looking at the edges leaving each of the nodes that correspond to states from which the assembly can be completed. The establishment conditions can also be obtained from the AND/OR graph of feasible assembly sequences by systematically looking at the hyperarcs of each assembly tree.

As an example, the establishment conditions for the assembly shown in figure 1 are:

$$F_1(x_1,x_2,x_3,x_4,x_5) = \bar{x}_1\cdot\bar{x}_2\cdot\bar{x}_3\cdot\bar{x}_4\cdot\bar{x}_5 + \bar{x}_1\cdot x_2\cdot\bar{x}_3\cdot\bar{x}_4\cdot\bar{x}_5 + \bar{x}_1\cdot\bar{x}_2\cdot x_3\cdot\bar{x}_4\cdot\bar{x}_5 + \bar{x}_1\cdot\bar{x}_2\cdot\bar{x}_3\cdot\bar{x}_4\cdot x_5 + \bar{x}_1\cdot x_2\cdot\bar{x}_3\cdot x_4\cdot\bar{x}_5 + \bar{x}_1\cdot\bar{x}_2\cdot x_3\cdot x_4\cdot x_5$$

$$F_2(x_1,x_2,x_3,x_4,x_5) = \bar{x}_1\cdot\bar{x}_2\cdot\bar{x}_3\cdot\bar{x}_4\cdot\bar{x}_5 + x_1\cdot\bar{x}_2\cdot\bar{x}_3\cdot\bar{x}_4\cdot\bar{x}_5 + \bar{x}_1\cdot\bar{x}_2\cdot x_3\cdot\bar{x}_4\cdot\bar{x}_5 + \bar{x}_1\cdot\bar{x}_2\cdot\bar{x}_3\cdot x_4\cdot\bar{x}_5 + x_1\cdot\bar{x}_2\cdot\bar{x}_3\cdot\bar{x}_4\cdot x_5 + \bar{x}_1\cdot\bar{x}_2\cdot x_3\cdot x_4\cdot x_5$$

$$F_3(x_1,x_2,x_3,x_4,x_5) = \bar{x}_1\cdot\bar{x}_2\cdot\bar{x}_3\cdot\bar{x}_4\cdot\bar{x}_5 + x_1\cdot\bar{x}_2\cdot\bar{x}_3\cdot\bar{x}_4\cdot\bar{x}_5 + \bar{x}_1\cdot x_2\cdot\bar{x}_3\cdot\bar{x}_4\cdot\bar{x}_5 + \bar{x}_1\cdot\bar{x}_2\cdot\bar{x}_3\cdot x_4\cdot\bar{x}_5 + \bar{x}_1\cdot\bar{x}_2\cdot\bar{x}_3\cdot\bar{x}_4\cdot x_5 + \bar{x}_1\cdot x_2\cdot\bar{x}_3\cdot x_4\cdot\bar{x}_5 + x_1\cdot\bar{x}_2\cdot\bar{x}_3\cdot\bar{x}_4\cdot x_5$$

$$F_4(x_1,x_2,x_3,x_4,x_5) = \bar{x}_1\cdot\bar{x}_2\cdot\bar{x}_3\cdot\bar{x}_4\cdot\bar{x}_5 + \bar{x}_1\cdot x_2\cdot\bar{x}_3\cdot\bar{x}_4\cdot\bar{x}_5 + \bar{x}_1\cdot\bar{x}_2\cdot x_3\cdot\bar{x}_4\cdot\bar{x}_5 + \bar{x}_1\cdot\bar{x}_2\cdot\bar{x}_3\cdot\bar{x}_4\cdot x_5 + x_1\cdot\bar{x}_2\cdot\bar{x}_3\cdot\bar{x}_4\cdot x_5 + x_1\cdot x_2\cdot x_3\cdot\bar{x}_4\cdot\bar{x}_5$$

$$F_5(x_1,x_2,x_3,x_4,x_5) = \bar{x}_1\cdot\bar{x}_2\cdot\bar{x}_3\cdot\bar{x}_4\cdot\bar{x}_5 + x_1\cdot\bar{x}_2\cdot\bar{x}_3\cdot\bar{x}_4\cdot\bar{x}_5 + \bar{x}_1\cdot\bar{x}_2\cdot x_3\cdot\bar{x}_4\cdot\bar{x}_5 + \bar{x}_1\cdot\bar{x}_2\cdot\bar{x}_3\cdot x_4\cdot\bar{x}_5 + \bar{x}_1\cdot x_2\cdot\bar{x}_3\cdot x_4\cdot\bar{x}_5 + x_1\cdot x_2\cdot x_3\cdot\bar{x}_4\cdot\bar{x}_5$$

The first establishment condition (F_1 (x_1,x_2,x_3 ,x_4,x_5)) corresponds to the fact thai the only states in which connection C_1 (i.e. the connection between the cap and the stick) can be established without precluding the completion of the assembly are either the state in which no connection has been established (node 1 in figure 3), or the state in which only connection c_2 is established (node 2), or the state in which only connection c_3 is established (node 4), or the state in which only connection c_5 is established (node 5), or the state in which only connections C_2 and C_4 are established (node 9), or the state in which only connection C_1 and C_2 are not established (node 12). It should be noticed that there is no term corresponding to the state in which only connection C_4 is established (node 6); although it is feasible to establish connection C_1, the resulting state (node 10) is a dead-end from which the assembly cannot be completed. Using the rules of boolean algebra the first establishment condition can be rewritten as

$$F_1(x_1,x_2,x_3,x_4,x_5) = \bar{x}_1\cdot\bar{x}_2\cdot\bar{x}_3\cdot\bar{x}_4 + \bar{x}_1\cdot\bar{x}_2\cdot\bar{x}_4\cdot\bar{x}_5 + \bar{x}_1\cdot\bar{x}_2\cdot x_3\cdot x_4\cdot x_5 + \bar{x}_1\cdot x_2\cdot\bar{x}_3\cdot\bar{x}_5$$

Although further simplification is still possible, in this paper it is assumed that a boolean function even when simplified is a sum of products. Depending on the application it may be useful to drop this restriction and to further simplify the boolean functions.

Any assembly sequence (X_1,X_2,,.....,X_N) that is feasible is such that if the i^{th} connection is established in task T_k then $F_i(X_k) = T$. And any assembly sequence is feasible if for any of all its tasks T_k are such that if the i^{th} connection is established in T_k then $F_i(X_k) = T$. Therefore, the set of establishment conditions is a correct and complete representation of assembly sequences.

The establishment conditions as defined in this paper can only discriminate between feasible and nonfeasible assembly sequences. There are sequences of states that "satisfy" the establishment conditions but are not assembly sequences and therefore cannot be feasible assembly sequences.

Knowing $F_1, F_2, \ldots\ldots F_L$, and the assembly's graph of connections, it is straight forward to construct the assembly's directed graph of assembly states.

6. Precedence Relationship Representation of Assembly Sequences

Two types of precedence relationships can be used to represent assembly sequences: precedence relationships between the establishment of one connection and the establishment of another connection, and precedence relationships between the establishment of one connection and states of the assembly process.

- Precedence relationships between the establishment of one connection and the establishment of another connection

We will use the notation $C_i < C_j$ to indicate the fact that the establishment of connection c_i must precede the establishment of connection c . And we will use the notation $c_i < c_j$ to indicate the fact that the establishment of connection c_i must precede or be simultaneous with the establishment of connection c•. Furthermore, we will use a compact notation for logical combinations of precedence relationships; for example, we will write $c_i < c_j\ c_k$ when we mean $(c_i < c_j) \wedge (C_l < c_k)$, and we will write $c_i + c_j < c_k$ when we mean $(c_i < c_k) \vee (c_j < c_k)$.

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 connection. For example, for the assembly shown in figure 1, the assembly sequence $x_1 = [F, F, F, F, F]$, $X_2 = [T, F, F, F, F]$, $x_3 = [T, T, T, F, F]$, and $x_4 = [T, T, T, T, T]$ can be uniquely characterized by the following conjunction of precedence relationships

$$(C_1 < C_2) \wedge (C_2 < C_4) \wedge (c_2 < r_3) \wedge (r_3 < r_2) \wedge (C_4 < C_5) \wedge (c_5 < c_4)$$

The set of all M feasible assembly sequences can be uniquely characterized by a disjunction of M conjunctions of precedence relationships in which each conjunction characterizes one assembly sequence. Clearly, this logical combination of precedence relationships constitutes a correct and complete representation for the set of all assembly sequences.

It is often possible to simplify this logical combination of precedence relationships using the rules of boolean algebra. Further simplification is possible if one notices that there are logical combinations of precedence relationships that cannot be satisfied by any assembly sequence. For the assembly shown in figure 1, for example, the combination $(C_1 < c_2) \wedge (c_2 < c_3) \wedge (c_3 < c_4) \wedge (c_4 < c_5)$ cannot be satisfied by any assembly sequence. These combinations can be set as don't care conditions in the simplification process.

For the assembly shown in figure 1, the precedence relationship

$$c_3 \leq c_1 \cdot c_5 + c_2 \cdot c_4 \equiv [(c_3 \leq c_1) \wedge (c_3 \leq c_5)] \vee [(c_3 \leq c_2) \wedge (c_3 \leq c_4)]$$

is sufficient to discriminate the feasible assembly sequences from the unfeasible assembly sequences.

Unlike the logical combination of precedence relationship obtained directly from the feasible assembly sequences, this logical combination of precedence relationships is satisfied by some sequences that are not assembly sequences. It can only discriminate the feasible assembly sequences from the unfeasible assembly sequences. This is due to the introduction of *don't care* conditions.

- Precedence relationships between the establishment of one connection and states of the assembly process

We will use the notation $c_i \longrightarrow S_k(x)$ to indicate that the establishment of the i^{th} connection must precede any state of the assembly process for which the value of the logical function $S_k(x)$ is T. The argument *of* $S_k(x)$ is an L~dimensional binary vector. We will also use a compact notation for logical combinations of precedence relationships. For example, we will write $C_i + C_j \rightarrow S_k(x)$ when we mean $[c_i \rightarrow S_k(x)] \vee [c_j \longrightarrow S_k(x)]$.

Let $\psi = \{ x_1, x_2, \bullet\bullet\bullet, j \}$ be a set that includes all unstable assembly states plus the stable states from which the final state cannot be reached plus the states that cannot be reached from the iniual state. Every element x of ψ is such that the value of the logical function $G(x_j)$ is T, where

$$G(\underline{x}) = G(x_1, x_2, \cdots, x_L) = \sum_{j=1}^{J} \prod_{l=1}^{L} \gamma_{jl}. \qquad \textit{(Eq. 1)}$$

The sum and the product in equation 1 are the logical operations OR and AND respectively, and y_{j1} is either the symbol X_1 if the l^{th} component of x is T, or the symbol x, if the 7^{th} component of x_j is F. In many cases the expression of $G(x)$ can be simplified using the rules of boolean algebra. Allowing for simplifications, but maintaining the assumption that a boolean function is a sum of products, equation 1 can be rewritten as

$$G(\underline{x}) = \sum_{j=1}^{J'} g_j(\underline{x}) \qquad \textit{(Eq.2)}$$

where $g_j(x)$ is the product of a subset of $\{ x_1, x_2, \ldots\ldots, x_L, x_1, x_2, \bullet\bullet\bullet, X_L \}$ that does not include both x_i and x_i for any i. Each term $g_j(x)$ can be rewritten grouping all the nonnegated variables first and all the negated variables last, i.e., $g_j(x) = x_a\, x_h\, x_b \ldots x_n\, x_p\, x_q \ldots x_z$.

Any assembly sequence that includes a state x_i for which $G(x_i) = T$ (i.e. $x_i \in \psi$) is not a feasible sequence. Therefore, a necessary condition for an assembly sequence $(x_1, x_2, \ldots\ldots x_N)$ to be feasible is that $G(x_1) = G(x2) = \ldots = G(x_N) = F$. This condition is equivalent to $g_j(x_i) = F$ for $i = 1, 2, \bullet\bullet\bullet, 7V$ and for $j = 1, 2, \ldots, /'$. For a special class of assemblies, this necessary condition is also sufficient. This class is composed of the assemblies that satisfy the following condition:

- If it is /easible to establish the i^{th} connection from a state in which the set of all connections that are established is T, then it is also feasible to establish the i^{th} connection from any state in which the set of all connections that are established is a subset of r.

If $C = \{ c_1, C_2, \ldots, c_L \}$ is the assembly's set of connections between parts and $(x_1, x_2, \ldots x_N)$ is an assembly sequence, the condition $g_j(x_1) = g_j(x_2) = \ldots = g_j(x_N) = F$ corresponds to a precedence relationship. The following theorem establishes the correspondence. A proof of this theorem is presented elsewhere [Homem de Mello 89].

Theorem 6: If $g(\underline{x}) = x_a \cdot x_b \cdot \cdots \cdot x_h \cdot \bar{x}_p \cdot \bar{x}_q \cdot \cdots \cdot \bar{x}_z$, $\{a, b, \cdots, h\} \cap \{p, q, \cdots, z\} = \emptyset$, $\{a, b, \cdots, h\} \cup \{p, q, \cdots, z\} \subseteq \{1, 2, \cdots, L\}$, and the sequence of L-dimensional binary vectors $(\underline{x}_1, \underline{x}_2, \cdots, \underline{x}_N)$ is such that $\underline{x}_1 = [F, F, \cdots, F]$, $\underline{x}_N = [T, T, \cdots, T]$, and $x_{il} = T \Rightarrow x_{jl} = T$ for all $j > i$, then the condition $g(\underline{x}_1) = g(\underline{x}_2) = \cdots = g(\underline{x}_N) = F$ is equivalent to the condition

$$c_p + c_q + \cdots + c_z \rightarrow S(\underline{x})$$

where

$$S(\underline{x}) = \prod_{l=1}^{L} \lambda_l \text{ and } \lambda_l = \begin{cases} x_l \text{ if } l \in \{a, b, \cdots, h\} \\ T \text{ otherwise} \end{cases}$$

with the product being the logical operation AND.

Each of the J' terms on the right side of equation 2 leads to one precedence relationship between the establishment of one connection and states of the assembly process. As a consequence of theorem 6, for assemblies that satisfy the condition stated above, if an assembly sequence is such that all J' precedence relationships are satisfied, then that sequence is feasible, and conversely. Therefore, the set of J' precedence relationships is a correct and complete representation of the set of all feasible assembly sequences.

For the assembly shown in figure 1, which satisfies the condition stated above, and whose directed graph of assembly states is shown in figure 3, Ψ includes $\underline{x}_8 = [F, T, F, F, T]$ and $\underline{x}_{10} = [T, F, F, T, F]$. Therefore,

$$G(\underline{x}) = G(x_1, x_2, x_3, x_4, x_5) = \\ = \bar{x}_1 \cdot x_2 \cdot \bar{x}_3 \cdot \bar{x}_4 \cdot x_5 + x_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \cdot x_4 \cdot \bar{x}_5$$

In this case the expression of $G(x)$ cannot be further simplified; the precedence relationships are:

$$c_1 + c_3 + c_4 \rightarrow x_2 \cdot x_5 \qquad c_2 + c_3 + c_5 \rightarrow x_1 \cdot x_4 \qquad \textit{(Set 1)}$$

A simpler set of precedence relationships can be obtained if in the simplification of $G(\underline{x})$ we set the nonstate vectors as don't care conditions. For the assembly shown in figure 1, the set of precedence relationships

$$c_1 \rightarrow x_2 \cdot x_5 \qquad c_2 \rightarrow x_1 \cdot x_4 \qquad \textbf{(Set 2)}$$

was obtained in the same fashion as Set 1, except for setting the nonstate vectors as don't care conditions. Set 2 is simpler and yet equivalent to Set 1.

Both set 1 and set 2 of precedence relationships can only discriminate feasible from non-feasible assembly sequences.

7. Conclusion

Four types of representations for assembly sequences were reviewed: the directed graph of feasible assembly sequences, the AND/OR graph of feasible assembly sequences, the set of establishment conditions, and the set of precedence relationships. The mappings of one representation into the others have been established. The correctness and completeness of these representations have also been established. To the author's knowledge no previous precedence relationship representation of assembly sequences has been shown to be correct and complete.

In previous work [Homem de Mello and Sanderson 89], we have presented an algorithm for the generation of the AND/OR graph of feasible assembly sequences and a proof of its correctness and completeness. Depending on the application, other representations of assembly sequences might be preferred. The results presented in this paper allows us to construct the other three types of representation and to guarantee their correctness and completeness.

Furthermore, there has been other work on the generation of assembly sequences [Bourjault 84, De Fazio and Whitney 87]. To the author's knowledge the correctness and completeness of these other algorithms have not been established. The results presented in this paper allow the proof of the correctness and completeness of algorithms for the generation of assembly sequences that yield precedence relationships or establishment conditions.

Acknowledgements

We thank Thomas L. De Fazio and Daniel E. Whitney for helpful discussions, and James Darnell for making the software for simplification of boolean functions available to us. The responsibility for the paper, of course, remains with the authors.

This research was supported by 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.

References

[Blum et al. 70] M. Blum et al. *A Stability Test for Configurations of Blocks.* Artificial Intelligence Memo 188, Massachusetts Institute of Technology, Feb, 1970.

[Boneschanscher et al. 88] N. Boneschanscher et al. Subassembly Stability. In *Proceedings AAAI-88 Seventh National Conference on Artificial Intelligence,* pages 780-785. American Association for Artificial Intelligence, Morgan Kaufman, Aug, 1988.

[Bourjault 84] A. Bourjault. *Contribution a une Approche Methodologique de L' Assemblage Automatisi: Elaboration Automatique des Sequences Operatoires.* These d'Etat, Universite de Franche-Comte, Besancon, France, Nov, 1984.

[De Fazio and Whitney 87] T. L. De Fazio and D. E Whitney. Simplified Generation of All Mechanical Assembly Sequences. *IEEE Journal of Robotics and Automation* RA-3(6):640-658, Dec, 1987. See corrections on same journal, RA-4(6):705-708, Dec, 1988.

[Homem de Mello 89] L. S. Homem de Mello. *Task Sequence Planning for Robotic Assembly.* PhD Thesis, Carnegie Mellon University, 1989.

[Homem de Mello and Sanderson 86] L. S. Homem de Mello and A. C. Sanderson. AND/OR Graph Representation of Assembly Plans. In *Proceedings AAAI-86 Fifth National Conference on Artificial Intelligence,* pages 1113-1119. American Association for Artificial Intelligence, Morgan Kaufmann Publishers, 1986.

[Homem de Mello and Sanderson 88] L. S. Homem de Mello and A. C. Sanderson. Task Sequence Planning for Assembly. In *IMACS World Congress '88 on Scientific Computation.* Paris, Jul, 1988.

[Homem de Mello and Sanderson 89] L. S. Homem de Mello and A. C. Sanderson. A Correct and Complete Algorithm for the Generation of Mechanical Assembly Sequences. In *Proceedings of the 1989 IEEE International Conference on Robotics and Automation.* IEEE Computer Society Press, Washington D.C., May, 1989.