

Enumerating the Non-Isomorphic Assembly Configurations of Modular Robotic Systems

I-Ming Chen and Joel W. Burdick

Division of Engineering and Applied Science
California Institute of Technology
Pasadena, CA 91125

The authors

Noted

The authors

Abstract

A "modular" robotic system consists of joint and link modules that can be assembled in a variety of configurations to meet different or changing task requirements. However, due to typical symmetries in module design, different assembly configurations may lead to robotic structures which are kinematically identical, or isomorphic. This paper considers how to enumerate the non-isomorphic assembly configurations of a modular robotic system. We introduce an *Assembly Incidence Matrix* (AIM) to represent a modular robot assembly configuration. Then we use symmetries of the module geometry and graph isomorphisms to define an equivalence relation on the AIMs. Equivalent AIMs represent isomorphic robot assembly configurations. Based on this equivalence relation, we propose an algorithm to generate non-isomorphic assembly configurations of an n -link tree-like robot with different joint and link module types. Examples demonstrate that this method is a significant improvement over a brute force enumeration process.

1. Introduction

By a modular robotic system we mean one in which various sub-assemblies, at the level of links and joints, can be easily separated and reassembled into different configurations which are individually well suited to the diverse task requirements. Because of their potential adaptability, modular robotic systems have received significant attention recently. Several papers have been devoted to the mechanical design and construction of modular robotic systems [3,15]. Others have considered modular robot software and control architectures [11]. Prototype modular systems have actually been built and demonstrated, including the "Reconfigurable Modular Manipulator System" (RMMS) developed by Khosla and coworkers at CMU [9], and the several generations of the "Cellular robotic system" (CEBOT) developed by Fukuda and coworkers at Nagoya University [6]. For the purposes of this paper, it should be noted that the modular systems developed or proposed to date have several common mechanical and structural features: (1) they contain two basic structure elements: link and joint modules [3,6,9,15]; (2) they employ simple joint designs: most of them have only 1-DOF revolute joints and 1-DOF prismatic joints [3,6,9]; (3) for interchangeability, the link geometries possess certain symmetries [3,6,9]; and (4) for adaptability, the joints can be attached to a link in many different ways [3].

To automatically determine a sufficient or optimal arrangement of the system modules for a given task, one might try a "generate-and-test" procedure in which all possible assembly configurations of the modular set are generated, and then each assembly configuration is tested against the task requirements to determine its sufficiency or optimality. However, due to symmetries in module geometry and robot structural topology, many different assembly configurations will have the same kinematic properties. Thus, a brute force enumeration of all module assemblies will result in the generation

of many functionally identical candidate structures. This is undesirable from a computational complexity point of view, as it leads to many unnecessary test procedures.

This paper proposes a systematic methodology to enumerate the non-isomorphic assembly configurations of a set of modules. This method is based on the symmetry properties of the modules and a graph representation of the robot's structural topology. We introduce an *Assembly Incidence Matrix* (AIM) to represent a robot assembly configuration and its associated kinematic graph. Equivalence relationships are defined on the AIMs using graph isomorphisms and the symmetric rotation group of individual link modules. AIMs in the same equivalence class represent isomorphic robots. The ensuing examples show that our procedure can significantly reduce the computational complexity of the enumeration process. In addition, this method is also useful when designing a modular robotic system, as it can answer the important question: "what is the set of uniquely different robots that I can construct from a given set of modules?"

2. Simplified Module Models

This paper concentrates on the problem of enumerating modular robotic assembly configurations, and not on practical, but important, mechanical design and control issues. Thus, we introduce a set of link and joint modules for the purposes of illustration. Any other set of modular links and joints can be analyzed in exactly the same way using our method.

2.1. Joint Modules

- Revolute joint (R): unlimited 1-DOF rotary motion between two link modules.
- Helical joint (H): 1-DOF twisting motion between two links.
- Cylindrical joint (C): 2-DOF motion between connected links: one is rotation, the other is translation along the rotation axis.

2.2. Link Modules

In a conventional robot, a link supports two joints at either end. In the modular robot case, more than two joints may be attached to a link module via numerous "connecting ports." The multiplicity of connecting ports may further allow different joint attachment positions and orientations. Based on those characteristics, we model a link as a rigid object whose geometric shape exhibits certain symmetries with respect to its shape and the distribution of its connecting ports. We define a *module coordinate system*, O , whose origin is located at the link's center of symmetry for each module.

- Square prism (L): The connecting ports are located symmetrically on each face as shown in Fig. 1. Any combination of 10 identical or mixed type joints can be simultaneously attached to

one of these links. The 10 connectors are labeled from 1 to 10 as shown in the figure. The origin of O is located at the prism's center of symmetry.

- **Cubic box (B):** This module has one connector on each of its 6 faces. The origin of O is located at the center of the cube.

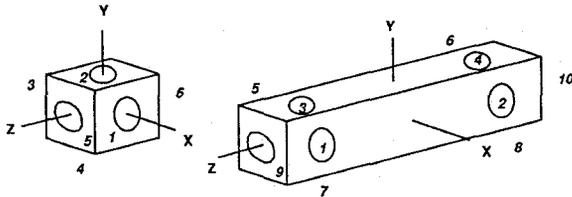


Fig 1. Link modules—a cubic box (B) and a prism (L)

3. Link-Joint Assembly Enumeration

In this section we study the symmetries and permutations which arise in connecting a variety of joints to one link. This is a necessary prelude to our analysis of multi-link assemblies. Suppose we wish to connect an R-joint and an H-joint to two of the 10 ports of a prism link module. There are $10 \cdot 9 = 90$ possible combinations and Fig. 2 shows three of them. If we consider geometric symmetry of the prism and neglect the labels on those ports, assembly states (a) and (b) are indistinguishable if rotations of the link by 90° along the module z-axis are allowed. When this link-joint assembly is put in a modular robot structure, (a) and (b) function identically because they have similar joint locations. We call such link-joint assemblies *equivalent*. We are interested in determining the distinct, or nonequivalent, link-joint assemblies, as they lead to different functionality and kinematic properties of the entire robot. In the above example, there are only 12 distinct assemblies.

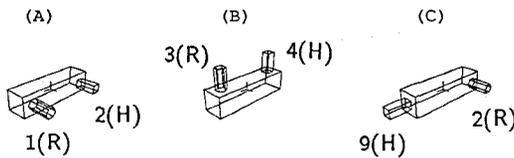


Fig 2. Three assembly configurations

Finding all distinct link-joint assembly configurations is similar to the cube coloring problem stated in [8]. The number of distinct configurations can be easily obtained from Pólya's theorem [1,8]; however, listing those configurations still requires an explicit algorithm. This section briefly reviews the process to solve this problem. For a more complete exposition, please refer to [1,7,8].

3.1. Symmetric Rotations and Permutations

SYMMETRIC ROTATIONS

Denote a link module by $\mathcal{L} \in \mathbb{R}^3$. A *symmetric rotation* $\varphi : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ of \mathcal{L} maps \mathcal{L} to itself, i.e., $\varphi(\mathcal{L}) = \mathcal{L}$. For example, rotations about a cubic or prismatic link module z-axis by 90° , 180° , or 270° are all symmetric rotations. The number of symmetric rotations of \mathcal{L} is finite because it is a symmetric polyhedron. The set of these rotations form a subgroup of $SO(3)$ called the *symmetric rotation group* of \mathcal{L} and is denoted by \mathcal{R} :

$$\mathcal{R} = \{\varphi \in SO(3) | \varphi(\mathcal{L}) = \mathcal{L}\}. \quad (3.1)$$

Recall that each link module connecting port is assigned a unique index. While a symmetric rotation $\varphi \in \mathcal{R}$ does not alter the position of the link, it does change the connecting port locations. One can imagine that the port locations after the rotation are a permutation of the indices before the rotation operation. As shown in

Fig. 3, the rotation of the prism link module about its z-axis by 90° causes port 1 to move to where port 3 was, port 2 to port 4, port 7 to port 1, etc.. Port 9 and 10 remain the same. This action can be written as a permutation

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 3 & 4 & 5 & 6 & 7 & 8 & 1 & 2 & 9 & 10 \end{pmatrix}. \quad (3.2)$$

Or in short, $\pi = (3,4,5,6,7,8,1,2,9,10)$. Suppose there are n ports on a link and we label them from 1 to n . Let $\text{PORT} = \{1, \dots, n\}$ be a set of integers containing all port numbers on a link module. A symmetric rotation $\varphi \in \mathcal{R}$ corresponds to a permutation $\pi : \text{PORT} \rightarrow \text{PORT}$. It can be shown that all permutations which corresponds to symmetric rotations on a link form a permutation group on PORT and we denote it by \mathcal{S} .

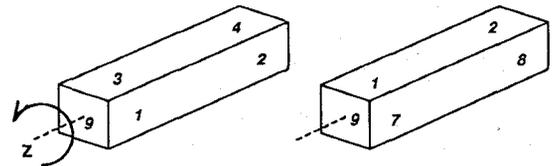


Fig 3. Rotating about z-axis 90°

THEOREM 3.1: (Cayley) [7] Every group with finite elements is isomorphic to a subgroup of a permutation group on a set of n elements for some integer n .

By theorem 3.1, \mathcal{R} is isomorphic to \mathcal{S} . Hence, we use π to represent either a permutation on PORT or a symmetric rotation on \mathcal{L} .

Let $\text{PORT}(L) = \{1, 2, \dots, 10\}$ and $\text{PORT}(B) = \{1, 2, \dots, 6\}$ be the sets of port numbers on prismatic and cubic links respectively. Denoting \mathcal{R}_L and \mathcal{R}_B as the symmetric rotation groups on prismatic and cubic links, and \mathcal{S}_L and \mathcal{S}_B as the permutation groups on $\text{PORT}(L)$ and $\text{PORT}(B)$, we get $\mathcal{R}_L \approx \mathcal{S}_L$ and $\mathcal{R}_B \approx \mathcal{S}_B$. Table 1 lists \mathcal{R}_L and \mathcal{S}_L . We can derive a similar table for \mathcal{R}_B and \mathcal{S}_B .

Rotations Axis Angle	Permutations on I_L										Type
	1	2	3	4	5	6	7	8	9	10	
identity	1	2	3	4	5	6	7	8	9	10	{10,0,0,0,0,0,0,0,0}
k 90°	3	4	5	6	7	8	1	2	9	10	{2,0,0,2,0,0,0,0,0}
k 180°	5	6	7	8	1	2	3	4	9	10	{2,0,0,4,0,0,0,0,0}
k 270°	7	8	1	2	3	4	5	6	9	10	{2,0,0,2,0,0,0,0,0}
i 180°	2	1	8	7	6	5	4	3	10	9	{0,5,0,0,0,0,0,0,0}
j 180°	6	5	4	3	2	1	8	7	10	9	{0,5,0,0,0,0,0,0,0}
i+j 180°	4	3	2	1	8	7	6	5	10	9	{0,5,0,0,0,0,0,0,0}
i-j 180°	8	7	6	5	4	3	2	1	10	9	{0,5,0,0,0,0,0,0,0}

Table 1. \mathcal{R}_L and \mathcal{S}_L

CYCLE INDEX OF A PERMUTATION GROUP

Note that a permutation $\pi : \text{PORT} \rightarrow \text{PORT}$ splits PORT uniquely into disjoint subsets called *cycles*, which contain elements of PORT cyclically permuted by π . A cycle is of length m if $\pi^m(s) = s \in \text{PORT}$ and $s, \pi(s), \dots, \pi^{m-1}(s)$ are all contained in this cycle. Let $n = |\text{PORT}|$, the total number of elements in PORT . We say π is of type $\{b_1, b_2, \dots, b_n\}$ if it splits PORT into b_1 cycles of length 1, b_2 cycles of length 2, and so on.

DEFINITION 3.2: *Cycle Index* [1,8,12] Let \mathcal{S} be a permutation group on PORT . The *cycle index* of \mathcal{S} is defined to be a polynomial in n dummy variables x_1, \dots, x_n as follows. For each $\pi \in \mathcal{S}$ of type $\{b_1, \dots, b_n\}$, form a product $x_1^{b_1} x_2^{b_2} \dots x_n^{b_n}$. The cycle index, $P_{\mathcal{S}}$, is defined to be

$$P_{\mathcal{S}}(x_1, \dots, x_n) = \frac{1}{|\mathcal{S}|} \sum_{\pi \in \mathcal{S}} x_1^{b_1} x_2^{b_2} \dots x_n^{b_n} \quad (3.3)$$

The cycle index of \mathcal{S}_L is

$$P_{\mathcal{S}_L}(x_1, x_2, \dots, x_{10}) = \frac{1}{8}(x_1^{10} + x_1^2 x_2^4 + 2x_1^2 x_4^2 + 4x_2^5) \quad (3.4)$$

3.2. Enumeration of Distinct Assembly Configurations

ASSEMBLY STATES

The physical action of connecting a variety of joints to a link module can be expressed as an injection mapping, $f : \text{PORT} \rightarrow \text{ATT}$, where ATT is the set of available joint types. In general, $\text{ATT} = \{0, R, H, C\}$ for our representative joint module set. Every port on the link is assigned an element in ATT corresponding to the type of joint inserted in the port. A zero signifies an empty port. We call this mapping, f , a link *assembly state*. For the example of Fig. 2, every one of the 90 combinations can be expressed as an assembly state $f : \{1, \dots, 10\} \rightarrow \{0, R, H\}$. Table 2 shows the assembly states corresponding to Fig. 2. In general, for an n -port link, k types of joints and d_i joints for each type (with $d_s = \sum_{i=1}^k d_i \leq n$), there are $\frac{n!}{d_1! \dots d_k! (n-d_s)!}$ possible assembly states. We denote the set of assembly states by \mathcal{F} .

States	Ports									
	1	2	3	4	5	6	7	8	9	10
f_a	R	H	0	0	0	0	0	0	0	0
f_b	0	0	R	H	0	0	0	0	0	0
f_c	0	R	0	0	0	0	0	0	H	0

Table 2. Three assembly states

Since many states in \mathcal{F} result in physically equivalent assembly configurations, we define an equivalence relation on assembly states in order to distinguish between inequivalent ones.

DEFINITION 3.3: Two assembly states $f_i, f_j : \text{PORT} \rightarrow \text{ATT}$, $f_i, f_j \in \mathcal{F}$, are *equivalent* iff there exists a symmetric rotation, $\pi \in \mathcal{S} \approx \mathcal{R}$, of the link module that transforms f_i to f_j , or visa versa:

$$f_i = f_j \circ \pi \quad (3.5)$$

Note that $\pi^{-1} \in \mathcal{S}$ by the group property and $f_i \circ \pi^{-1} = f_j$. Continuing our example, let π represent the permutation associated with the rotation of the prism about its z-axis by 90° . For the assembly states f_a and f_b in Table 2, we have $f_a(1) = f_b \circ \pi(1) = f_b(3) = R$, $f_a(2) = f_b \circ \pi(2) = f_b(4) = H$, and $f_a(k) = f_b \circ \pi(k) = 0$ for $k = 3, 4, \dots, 10$. Hence, $f_a = f_b \circ \pi$.

This equivalence relation divides \mathcal{F} into disjoint subsets, termed *equivalence classes* or *orbits*, under the action of the symmetric rotation group $\mathcal{R} \approx \mathcal{S}$ [7]. States in the same equivalence class are equivalent; those in different ones are not. Thus, the problem of enumerating the distinct link-joint assemblies is equivalent to finding the number of orbits of \mathcal{F} under \mathcal{S} . We denote the set of these orbits by \mathcal{F}/\mathcal{S} .

ORBITAL ENUMERATION—PÓLYA THEOREM

THEOREM 3.4: (Pólya Theorem)[1,8] Suppose \mathcal{S} is a permutation group on PORT isomorphic to \mathcal{R} and $P_{\mathcal{S}}$ is the cycle index of \mathcal{S} . Assign a dummy variable y_i to every element in ATT, e.g., y_0 to 0, y_1 to Type-1 joint, y_2 to Type-2 joint. The *inventory* of orbits, I_P , [8] in \mathcal{F}/\mathcal{S} can be derived by substituting x_k in $P_{\mathcal{S}}$ with $\sum y_i^{k_i}$, i.e.,

$$I_P = P_{\mathcal{S}}(\sum y_i, \sum y_i^2, \dots, \sum y_i^n), \quad (3.6)$$

where $n = |\text{PORT}|$. The coefficient of term $y_0^{d_0} y_1^{d_1} y_2^{d_2} \dots$ in (3.6) indicates the number of distinct link-joint assemblies (or orbits) with d_0 empty ports, d_1 Type-1 joints, d_2 Type-2 joints, etc.

EXAMPLE 3.5: For the example of Fig. 2, the assembly state can be written as $f : \text{PORT}(L) \rightarrow \{0, R, H\}$. Now assign y_0 to 0, y_1 to R-joints, and y_2 to H-joints. According to Theorem 3.4, the inventory of orbits is

$$\begin{aligned} I_P &= P_{\mathcal{S}_L}(x_1, x_2, \dots, x_n) = P_{\mathcal{S}_L}(x_1, x_2, x_4) \\ &= P_{\mathcal{S}_L}(y_0 + y_1 + y_2, y_0^2 + y_1^2 + y_2^2, y_0^4 + y_1^4 + y_2^4) \\ &= y_0^{10} + 2y_0^9 y_1 + \dots + 12y_0^8 y_1 y_2 + 106y_0^6 y_1^3 y_2 + \dots \end{aligned} \quad (3.7)$$

The number of distinct assembly states with 1 R-joint and 1 H-joint is 12, which is the coefficient of the $y_0^8 y_1 y_2$ term in (3.7). A brute force enumeration of the possible assembly states which does not account for the symmetry results in $10 \cdot 9 = 90$ assembly states! Similarly, the coefficient of term $y_0^6 y_1^3 y_2$ indicates that there are 106 unique ways to attach 3 R-joints and 1 H-joint to a prism link. Brute force enumeration would result in $\frac{10!}{6!3!1!} = 840$ different, but not unique, assembly states. ■

ORBITAL ENUMERATION—ALGORITHMIC APPROACH

While the number of these orbits can be easily obtained from Theorem 3.4, listing them requires an algorithm. Our algorithm `OrbitEnumerate` generates orbits of an assembly state set under the action of a symmetric rotation group. `OrbitEnumerate` accepts two arguments: an assembly state set, \mathcal{F} , and a symmetric rotation group, \mathcal{R} , written in permutation group form. If the number and type of joints are specified, all elements of input state set \mathcal{F} for an n -port link can be found using the standard *backtrack* algorithm [10]. The output of `OrbitEnumerate` is a list of distinct assembly states. Each state represents an orbit of \mathcal{F} . Let $\mathcal{F} = \{f_1, \dots, f_{|\mathcal{F}|}\}$ and $\mathcal{R} = \{\pi_1, \dots, \pi_m\}$, where $|\mathcal{F}|$ is the number of possible assembly states. The algorithm works as follows.

```

0  Procedure OrbitEnumerate( $\mathcal{R}, \mathcal{F}$ )
1  Queue = {1, 2, ...,  $|\mathcal{F}|$ };
2  NewOrbit =  $\emptyset$ ;
3  While Length(Queue) > 1 do
4  {
5    v = First(Queue);
6    Queue = Rest(Queue);
7    VOrbit =  $\emptyset$ ;
8    For all  $\pi_i \in \mathcal{R}$  do
9      { Append( $f_v \circ \pi_i$ , VOrbit) };
10   tmp = Queue;
11   For all  $i \in \text{tmp}$  do
12     { If  $f_i \in \text{VOrbit}$  then Queue = Delete( $i$ , Queue) };
13   Append( $f_v$ , NewOrbit);
14   };
15 If Queue = { $k$ } then Append( $k$ , NewOrbit);
16 Return(NewOrbit);

```

From experiment, we observed that `OrbitEnumerate` finds orbits of \mathcal{F} under \mathcal{R} in $O(N_{eq}^2)$ time for a fixed \mathcal{R} , where N_{eq} is the number of orbits. N_{eq} can be determined from the coefficients of I_P using Theorem 3.4 when the number and types of joints are specified. The "orderly algorithm" of Williamson [12] can also be used to list the orbits of a set under a group action.

4. Representations of Modular Robots

We use a *kinematic graph* to portray the connection between links and joints in a modular robot. In a kinematic graph, vertices represent links while edges represent joints. This technique is often used in mechanism design to represent the structure of kinematic chains [5,13]. We consider two classes of robots: *homogeneous* modular robots (multiple copies of a single type of joint and link module), and *hybrid* modular robots (different types of joints and links). We first discuss the homogeneous modular robot case in detail. The analysis of hybrid robots follows naturally.

First, we introduce some graph terminologies used in the sequel. A graph $G = (V, E)$ consists of a vertex set, V , and an edge set, E , such that every edge in E is associated with a pair of vertices. A *labelled* graph has vertices labelled v_1, v_2 , etc. and edges labelled e_1, e_2 , etc. Suppose G is labelled and consists of m vertices and n edges, then $V = \{v_1, \dots, v_m\}$ and $E = \{e_1, \dots, e_n\}$. Because we are concerned with how links and joints are connected together, we represent G by an $m \times n$ *vertex-edge incidence matrix* denoted $M(G)$. $m_{ij} = 1$ if edge e_j is incident on vertex v_i ; $m_{ij} = 0$, otherwise.

Two labelled graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are *isomorphic* if there is a bijective mapping γ_v from V_1 to V_2 , and a bijective mapping γ_e from E_1 to E_2 . That is, there exists a 1-to-1 correspondence between their vertex and edge sets that conserves the incidence relations. We call $\gamma_{12} = (\gamma_v, \gamma_e)$ an isomorphism from G_1 to G_2 . If $V_1 = V_2$ and $E_1 = E_2$, γ_v and γ_e can be thought as permutations on V_1 and E_1 , or equivalently, row and column permutations on $M(G_1)$ respectively. Let $M_\gamma(G_1)$ denote the incidence matrix of G_1 after the permutation $\gamma = (\gamma_v, \gamma_e)$. The incidence matrices of isomorphic labelled graphs differ by permutation of columns and rows only [4]. Suppose γ_{12} is an isomorphism from G_1 to G_2 , then $M_{\gamma_{12}}(G_1) = M(G_2)$. Fig. 4 shows 3 isomorphic graphs. The isomorphism from (a) to (b) is $\gamma_{ab} = (\gamma_{v1}, \gamma_{e1}) = ((2, 1, 4, 3), (c, b, a))$; the one from (a) to (c) is $\gamma_{ac} = ((1, 2, 4, 3), (a, c, b))$.

If a labelled graph, G , exhibits geometric symmetry, there exists isomorphisms of G to itself, i.e., there exists γ such that $M_\gamma(G) = M(G)$. E.g., graphs in Fig.4 (a) and (c) are mirror images with respect to edge a . Their incidence matrices are identical. We call such isomorphism an *automorphism*. These automorphisms forms a group called the *automorphism group* of G , denoted by $\mathcal{H}(G)$. The automorphism group of the graph in Fig. 4(a) contains 6 elements:

$$\begin{aligned} &((1, 2, 3, 4), (a, b, c)) \quad ((1, 3, 4, 2), (b, c, a)) \quad ((1, 4, 2, 3), (c, a, b)) \\ &((1, 2, 4, 3), (a, c, b)) \quad ((1, 4, 3, 2), (c, b, a)) \quad ((1, 3, 2, 4), (b, a, c)) \end{aligned}$$

where $((1, 2, 3, 4), (a, b, c))$ is the identity element. Note that the automorphism group of an asymmetric graph contains the identity element only.

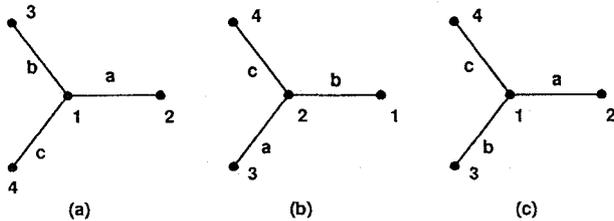


Fig 4. Isomorphism and automorphism of graphs

4.1. Homogeneous Modular Robots

Since there is only one type of link and joint module in a homogeneous modular robot, we can use a labelled kinematic graph to represent its structural topology. The kinematic graph, G , of an m -link and n -joint homogeneous modular robot is a labelled graph with m vertices and n edges. Fig. 5 shows a particular assembly of 4 prism link modules and 3 revolute joint modules. Fig.6 shows its associated labelled kinematic graph.

To fully describe a modular robot construction, one must assign port information to the links for every joint. Unfortunately, the standard incidence matrix can not encode connecting port information. To alleviate this problem, we introduce an *assembly incidence matrix* (AIM), which resembles an incidence matrix in that it mod-

els the connectivity of links and joints, but also shows connecting port information.

DEFINITION 4.1: The assembly incidence matrix, $A(G)$, of a modular robot kinematic graph, G , is obtained by replacing every entry of 1 in $M(G)$ with a non-zero integer, $k \in \text{PORT}$, which corresponds to the connecting port label. The zero entries are unchanged. Thus, every nonzero entry $a_{ij} = k$ indicates that joint e_j is attached to port k of link v_i . $a_{ij} = 0$, otherwise.

The i^{th} row of $M(G)$ indicates how joints are connected to link v_i ; the j^{th} column shows how two adjacent links are connected by joint e_j . Therefore, an AIM completely determines the construction of a modular robot, and a robot can be built from a set of modules according to its AIM without ambiguity.

EXAMPLE 4.2: Fig. 5 shows a modular robot constructed from 4 prism and 3 revolute joint modules. Its labelled kinematic graph, G_a , contains 4 vertices (Fig. 6). The incidence matrix and AIM for this assembly configuration are

$$M(G_a) = \begin{matrix} & e_1 & e_2 & e_3 \\ v_1 & \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{matrix} \quad A(G_a) = \begin{matrix} & e_1 & e_2 & e_3 \\ v_1 & \begin{pmatrix} 10 & 5 & 1 \\ 0 & 2 & 0 \\ 9 & 0 & 0 \\ 0 & 0 & 4 \end{pmatrix} \end{matrix} \quad (4.1)$$

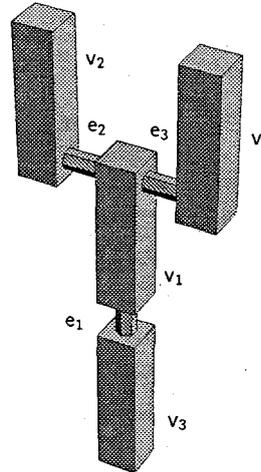


Fig 5. A homo. robot

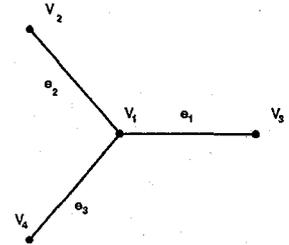


Fig 6. G_a

Our major concern is the functionality of the final robot construction, i.e., the topology of the robot, the joint positions on the links, and its kinematic properties. While it is true that a robot can be uniquely constructed from a given AIM, it is not necessarily true that different AIMs lead to functionally different robots. Different AIMs may result in functionally identical modular robot constructions because of: (1) isomorphisms on labelled kinematic graphs; and (2) link module symmetries. For example, consider the following AIMs which generate robot constructions having identical kinematic function to $A(G_a)$ in Example 4.2:

$$A(G_b) = \begin{matrix} & e_1 & e_2 & e_3 \\ v_1 & \begin{pmatrix} 2 & 0 & 0 \\ 5 & 10 & 1 \\ 0 & 9 & 0 \\ 0 & 0 & 4 \end{pmatrix} \end{matrix} \quad A(G_c) = \begin{matrix} & e_1 & e_2 & e_3 \\ v_1 & \begin{pmatrix} 10 & 5 & 1 \\ 0 & 2 & 0 \\ 10 & 0 & 0 \\ 0 & 0 & 4 \end{pmatrix} \end{matrix}$$

$$A(G_d) = \begin{matrix} & e_1 & e_2 & e_3 \\ \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{matrix} & \begin{pmatrix} 10 & 1 & 5 \\ 0 & 4 & 0 \\ 9 & 0 & 0 \\ 0 & 0 & 2 \end{pmatrix} \end{matrix}. \quad (4.2)$$

G_b is isomorphic to G_a under $\gamma_{ba} = ((v_2, v_1, v_3, v_4), (e_2, e_1, e_3))$. Note that $M(G_c) = M(G_d) = M(G_a)$; they are structurally identical.

Let us consider the first case: $M_{\gamma_{ba}}(G_b) = M(G_a)$ because γ_{ba} is an isomorphism from G_b to G_a . Row and column permutations of the incidence matrices corresponds to isomorphisms on the associated graphs. Permuting edge and vertex labels does not alter the robot structure or change the connecting ports. Thus, the values of non-zero AIM entries do not change during permutations. Let $A_{\gamma_{ba}}(G_b)$ denote the permuted $A(G_b)$ according to γ_{ba} , then $A_{\gamma_{ba}}(G_b) = A(G_a)$ —indicating that $A(G_a)$ and $A(G_b)$ represent identical modular robot constructions.

Secondly, the geometric symmetry of the link modules may allow a non-unique port assignment when a link is connected to the robot. For example, $A(G_c)$ differs from $A(G_a)$ in the (3,1) entry only. Recall from Section 3 that connecting e_1 to port 9 or to port 10 leads to two equivalent assembly states, since there is a symmetric rotation about the x-axis of prism v_3 , $\pi = (\dots, 10, 9)$, that transforms from the former state to the latter one. If we consider only the locations of joint e_2 on link v_3 , robots constructed by $A(G_c)$ and $A(G_a)$ are indistinguishable.

We thus say that $A(G_a)$, $A(G_b)$, and $A(G_c)$ are *equivalent*: each one can be transformed into the other via allowable graph isomorphisms or symmetric rotations of a link. Based on this example, it is clear that we must construct an equivalence relationship on AIMs, analogous to the equivalence relationship on the assembly states of a link. Let G_1 and G_2 be the labelled kinematics graphs of two robots, and $A(G_1)$ and $A(G_2)$ are their AIMs. The equivalence relationship is constructed in two steps.

STRUCTURAL EQUIVALENCE

DEFINITION 4.3: Two AIMs $A(G_i)$, $i = 1, 2$, are *structurally equivalent*, if and only if G_1 and G_2 are isomorphic.

In (4.2), $A(G_a)$, $A(G_b)$ and $A(G_c)$ are structurally equivalent.

LINK ASSEMBLY EQUIVALENCE

Two modular robot assemblies will function identically not only if they have the same topology, but also if the locations of joints on corresponding links are also matched. Suppose $A(G_1)$ and $A(G_2)$ are structurally equivalent. Let γ_{12} be the isomorphism from G_1 to G_2 . Denote \hat{w}_1^i and \hat{w}_2^i the i^{th} row vector of $A_{\gamma_{12}}(G_1)$ and $A(G_2)$ respectively.

DEFINITION 4.4: Two row vectors $\hat{w}_1^i = (a_{i1}^1, \dots, a_{in}^1)$, and $\hat{w}_2^i = (a_{i1}^2, \dots, a_{in}^2)$, are *link assembly equivalent*, if and only if there is a symmetric rotation $\pi : \text{PORT} \rightarrow \text{PORT}$ of link v_i such that (1) for non-zero entry, $a_{ij}^1 \in \text{PORT}$, $\pi(a_{ij}^1) = a_{ij}^2$; (2) for zero entry, $a_{ij}^1 = a_{ij}^2 = 0$.

If the i^{th} row vectors of two AIMs are link assembly equivalent, they represent functionally equivalent assemblies on link v_i . If the corresponding rows in $A_{\gamma_{12}}(G_1)$ and $A(G_2)$ are all joint assembly equivalent, $A(G_1)$ and $A(G_2)$ are equivalent. In the above example, rows 1, 2, and 4 of $A(G_a)$ and $A(G_c)$ are exactly equivalent. For $A(G_c)$, a rotation about the x-axis of prism v_3 by 180° transforms the 3^{rd} row, (10, 0, 0), into (9, 0, 0), the 3^{rd} row of $A(G_a)$, so

(10, 0, 0) \sim (9, 0, 0). Therefore, $A(G_a)$ and $A(G_c)$ are equivalent.

GRAPH AUTOMORPHISMS

If a graph G exhibits symmetry, every element in $\mathcal{H}(G)$ will render $A(G)$ similar to itself structurally. Hence, one also must consider all automorphisms of G_2 when comparing for the link assembly equivalence on rows of structurally equivalent AIMs, $A_{\gamma_{12}}(G_1)$ and $A(G_2)$, where γ_{12} is the isomorphism from G_1 to G_2 . If there exists an $\eta \in \mathcal{H}(G_2)$ which makes all corresponding rows of $A_{\gamma_{12}}(G_1)$ and $A_\eta(G_2)$ link assembly equivalent, then $A(G_1)$ and $A(G_2)$ are equivalent. For instance, $A(G_d)$ in (4.2) is structurally equivalent to $A(G_a)$. The automorphism $\eta = ((v_1, v_4, v_3, v_2), (e_1, e_3, e_2))$ of G_d makes $A_\eta(G_d) = A(G_a)$. This means that $A(G_d)$ and $A(G_a)$ actually represent equivalent constructions.

DEFINITION 4.5: Two AIMs $A(G_1)$ and $A(G_2)$ are equivalent if and only if they are structurally equivalent, i.e., G_1 and G_2 are isomorphic, and there exist an automorphism of G_2 , $\eta \in \mathcal{H}(G_2)$, such that all row vectors in $A_{\gamma_{12}}(G_1)$ and $A_\eta(G_2)$ are link assembly equivalent, where γ_{12} is an isomorphism from G_1 to G_2 .

According to this definition, $A(G_b)$, $A(G_c)$, $A(G_d)$ in (4.2) and $A(G_a)$ of example 4.2 are all equivalent AIMs. Two robots are said to be isomorphic iff their AIMs are equivalent. They have the same outward appearance and same kinematic properties such as workspace and singularities.

4.2. Hybrid Modular Robot

A labelled kinematic graph, G , can not represent a hybrid modular robot structure because the vertices and edges must represent different kinds of links and joints. We use a labelled kinematic graph with link and joint types assigned on all of its vertices and edges to overcome this deficiency. We call such a graph a *specialized graph* [16], and denote it by \mathcal{G} . The specialized graph can be represented by an *extended incidence matrix* as follows. Let LTYPE and JTYPE denote the set of available link and joint types, e.g., LTYPE = $\{L, B\}$ and JTYPE = $\{R, H, C\}$.

DEFINITION 4.6: Let \mathcal{G} be the specialized kinematic graph of a hybrid modular robot with m links and n joints. We label the vertices from v_1 to v_m with elements of LTYPE and edges e_1 to e_n with labels from JTYPE. The *extended incidence matrix*, $\mathcal{M}(\mathcal{G})$, is an $(m+1) \times (n+1)$ matrix such that:

1. $m_{ij} = 1$, if joint e_j is attached to link v_i , and $m_{ij} = 0$, otherwise. $i = 1, \dots, m$, $j = 1, \dots, n$.
2. $m_{i,n+1} \in \text{LTYPE}$ represents v_i 's link type, $i = 1, \dots, m$.
3. $m_{m+1,j} \in \text{JTYPE}$ represents e_j 's joint type, $j = 1, \dots, n$.
4. $m_{m+1,n+1} = 0$.

The upper-left $m \times n$ submatrix of $\mathcal{M}(\mathcal{G})$ is the incidence matrix of labelled-only graph G , containing the topology of the robot. The information regarding types of links and joints are kept in the last column and the last row of $\mathcal{M}(\mathcal{G})$.

Two specialized kinematic graphs, \mathcal{G}_1 and \mathcal{G}_2 , are isomorphic if their labelled-only kinematic graphs, G_1 and G_2 , are isomorphic AND if the module type assignments to vertices and edges of both graphs are matched. Similar to labelled graph case, this isomorphism is still defined on the labels of vertices and edges and denote it by $\gamma = (\gamma_v, \gamma_e)$. γ is equivalent to row and column permutations on the extended incidence matrix, but permutation actions take place in the first m rows and n columns of the matrix (those containing structural information). Let $\mathcal{M}_\gamma(\mathcal{G})$ denote the extended incidence matrix of \mathcal{G} after the permutation γ . If γ_{12} represent the

isomorphism from \mathcal{G}_1 to \mathcal{G}_2 , then $\mathcal{M}_{\gamma_{12}}(\mathcal{G}_1) = \mathcal{M}(\mathcal{G}_2)$.

An automorphism, η , of \mathcal{G} , will render $\mathcal{M}(\mathcal{G})$ similar to itself, i.e., $\mathcal{M}_\eta(\mathcal{G}) = \mathcal{M}(\mathcal{G})$. A specialized graph \mathcal{G} also has an automorphism group $\mathcal{H}(\mathcal{G})$ if it exhibits symmetry. This group is a subgroup of the automorphism group of its labelled-only graph G : $\mathcal{H}(\mathcal{G}) \subset \mathcal{H}(G)$. For example, the specialized graph in Fig. 8 has no symmetry. Its automorphism group contains only the identity element.

Similar to Definition 4.1, the AIM of a hybrid modular robot is obtained by replacing every entry of 1 with a non-zero integer $k \in \text{PORT}$, while keeping zero entries unchanged. We call this an *extended assembly incidence matrix* (eAIM) and denote it by $\mathcal{A}(\mathcal{G})$.

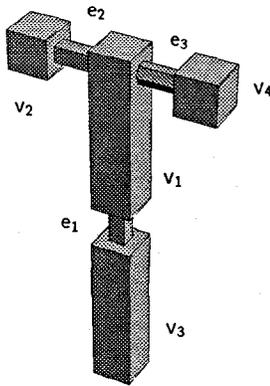


Fig 7. A hybrid robot

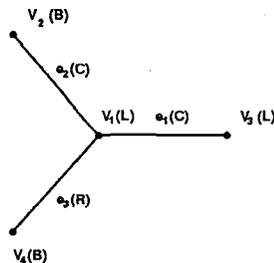


Fig 8. \mathcal{G}

EXAMPLE 4.7: Fig. 7 shows a hybrid robot built from 2 prisms, 2 cubes, 2 C-joints, and an R-joint. Its specialized kinematic graph is shown in Fig. 8. $\mathcal{M}(\mathcal{G})$ and $\mathcal{A}(\mathcal{G})$ are

$$\mathcal{M}(\mathcal{G}) = \begin{matrix} & e_1 & e_2 & e_3 & & \\ v_1 & 1 & 1 & 1 & L & \\ v_2 & 0 & 1 & 0 & B & \\ v_3 & 1 & 0 & 0 & L & \\ v_4 & 0 & 0 & 1 & B & \\ & C & C & R & 0 & \end{matrix} \quad \mathcal{A}(\mathcal{G}) = \begin{matrix} & e_1 & e_2 & e_3 & & \\ v_1 & 10 & 5 & 1 & L & \\ v_2 & 0 & 1 & 0 & B & \\ v_3 & 9 & 0 & 0 & L & \\ v_4 & 0 & 0 & 2 & B & \\ & C & C & R & 0 & \end{matrix} \quad (4.3)$$

EAIM EQUIVALENCE

A hybrid modular robot can be constructed from its eAIM without ambiguity. However, different eAIMs may lead to functionally identical hybrid robot constructions due to isomorphisms between specialized graphs and link module symmetries. As in the homogeneous case, we similarly define an equivalence relation on eAIMs based on specialized graph structural equivalence and link assembly equivalence. Let \mathcal{G}_1 and \mathcal{G}_2 be the specialized kinematic graphs of two hybrid robots, and $\mathcal{A}(\mathcal{G}_1)$ and $\mathcal{A}(\mathcal{G}_2)$ are their eAIMs.

DEFINITION 4.8: Two eAIMs $\mathcal{A}(\mathcal{G}_i)$, $i = 1, 2$, are *structurally equivalent*, if and only if the specialized graphs \mathcal{G}_1 and \mathcal{G}_2 are isomorphic.

Suppose γ_{12} is an isomorphism from \mathcal{G}_1 to \mathcal{G}_2 , then $\mathcal{M}_{\gamma_{12}}(\mathcal{G}_1) = \mathcal{M}(\mathcal{G}_2)$. Let \hat{w}_1^i and \hat{w}_2^i be the i^{th} row vectors of the upper-left $m \times n$ submatrix of $\mathcal{A}_{\gamma_{12}}(\mathcal{G}_1)$ and $\mathcal{A}(\mathcal{G}_2)$ respectively. The definition of link assembly equivalence on \hat{w}_1^i and \hat{w}_2^i is similar to Definition 4.4, with the understanding that the symmetric rotation action depends upon the type of link module. $\mathcal{A}(\mathcal{G}_1)$ and $\mathcal{A}(\mathcal{G}_2)$ are equivalent if all corresponding rows in $\mathcal{A}_{\gamma_{12}}(\mathcal{G}_1)$ and $\mathcal{A}(\mathcal{G}_2)$ are link assembly equivalent. Likewise, when checking for link assembly equivalence on rows of structurally equivalent eAIMs, $\mathcal{A}_{\gamma_{12}}(\mathcal{G}_1)$ and $\mathcal{A}(\mathcal{G}_2)$, one has to consider the automorphisms of \mathcal{G}_2 .

DEFINITION 4.9: Two eAIMs $\mathcal{A}(\mathcal{G}_1)$ and $\mathcal{A}(\mathcal{G}_2)$ are equivalent if and only if they are structurally equivalent, and there exist an automorphism of \mathcal{G}_2 , $\eta \in \mathcal{H}(\mathcal{G}_2)$ such that all row vectors in $\mathcal{A}_{\gamma_{12}}(\mathcal{G}_1)$ and $\mathcal{A}_\eta(\mathcal{G}_2)$ are link assembly equivalent, where γ_{12} is the isomorphism from \mathcal{G}_1 to \mathcal{G}_2 .

By Definitions 4.5 and 4.9, we are able to compare two AIMs for their functional equivalence. More importantly, this equivalence relations serve as a basis for enumerating non-isomorphic robot assembly configurations from a given set of modules.

5. Modular Robot Assembly Enumeration

We now apply the previous methods to develop the main practical result of this paper: a method to enumerate all of the non-isomorphic n -link tree-like hybrid modular robots from a given set of link and joint modules. An n -link tree-like robot has $n - 1$ joints and contains no closed-loop constructions of links and joints. We consider only tree-like topologies, and not topologies with closed loops because closed-loop modular robot constructions require additional kinematic constraints.

We divide tree-like robots into two classes: *free flying* and *fixed base*. A free-flying robot does not have an identified base link, while a fixed base robot does. The robot's base can be considered as a different link type. Fixed base robots are thus treated as hybrid robots, with the base link location determined during the hybrid robot specialization process. Homogeneous robots are necessarily free-flying robots. Conversely, a free-flying robot may be either a homogeneous or hybrid robot.

5.1. The Enumeration Process

The enumeration process begins with a given link set, LINK, with n elements and a joint set, JOINT, with $n - 1$ elements. The output is non-isomorphic modular robot assembly configurations represented by inequivalent eAIMs. The details of this procedure follow.

STEP 1: Generate non-isomorphic unassigned trees $\{G_i\}$ with n vertices. Label these trees, and for each tree G_i generate the associated $n \times m$ incidence matrix form $M(G_i)$.

A rooted tree corresponds to a fixed base robot with the root vertex representing the fixed base. A free tree has no root, and corresponds to a free-flying robot. Beyer and Hedetniemi [2] introduced a constant time algorithm to generate all rooted trees of a give size (the number of vertices). Based on this work, Wright et al. [14] propose a constant time algorithm to generate all free trees of a given size. We need only free trees in this step.

STEP 2: For every G_i , find its automorphism group $\mathcal{H}(G_i)$ using the backtrack algorithm [10].

STEP 3: Find distinct assignments from LINK and JOINT to vertices and edges of G_i under the automorphism group $\mathcal{H}(G_i)$. From those distinct assignments, construct non-isomorphic specialized trees $\{\mathcal{G}_{ik}\}$ based on G_i and write them in extended incidence matrices $\mathcal{M}(\mathcal{G}_{ik})$.

An assignment from LINK and JOINT to the vertices and edges of G_i is a 1-to-1 and onto function which is similar to an assembly state on the link. The automorphism group $\mathcal{H}(G_i)$ due to the symmetry of G_i is a permutation group on labels of vertices and edges similar to the symmetry rotation group of the link. Hence, we apply OrbitEnumerate to find distinct module assignments on G_i . Yan and Hwang [16] proposed a heuristic algorithm based on the *chain group* of a kinematic chain to enumerate non-isomorphic specialized mechanism for a specific kinematic chain. A fixed base robot can be obtained by putting a base link in the LINK set. The

location of this base link in the kinematic graph is determined in this step.

STEP 4: For every non-isomorphic specialized tree \mathcal{G} (or $\mathcal{M}(\mathcal{G})$), do the following:

- Since $\mathcal{H}(\mathcal{G}) \subset \mathcal{H}(G)$, the automorphism group of its unassigned graph G , pick out $\eta \in \mathcal{H}(G)$ such that $\mathcal{M}_\eta(G) = \mathcal{M}(\mathcal{G})$ to form $\mathcal{H}(\mathcal{G})$.
 - Generate distinct assemblies for every link, based on the incident edges of every vertex in \mathcal{G} , i.e., a row vector in $\mathcal{M}(\mathcal{G})$. Write these link-joint assemblies in eAIM row vector form.
- Every vertex along with its incident edges in \mathcal{G} determines the number and type of joints attached to the corresponding link. We use OrbitEnumerate to generate distinct assemblies on every link with those prescribed joints. Since \mathcal{G} is labeled, the labeled joints are treated as different joint types when applying OrbitEnumerate. For example, the first row of $\mathcal{M}(\mathcal{G})$ in (4.3) is $(1, 1, 1, L)$, which indicates that C-joints e_1 and e_2 and R-joint e_3 are attached to prism link v_1 . Using OrbitEnumerate, we obtain 46 distinct assemblies for 3 different joints attached to a prism. Replacing 1's in $(1, 1, 1, L)$ with port numbers from the distinct assemblies, we get 46 inequivalent 1st row vector representations for an eAIM. Repeat this process for every row of $\mathcal{M}(\mathcal{G})$.
- Take combinations of all inequivalent eAIM row vector representations for every row of $\mathcal{M}(\mathcal{G})$ to construct inequivalent eAIMs, $\mathcal{A}(\mathcal{G})$.
 - If the specialized graph has no symmetry, then go to the next step. Otherwise, use $\mathcal{H}(\mathcal{G})$ to eliminate equivalent eAIMs due to graph symmetry.

STEP 5: Repeat Step 4 for all specialized trees, then stop.

REMARK: This procedure can be applied to homogeneous robots by omitting the intermediate specialization process.

5.2. Examples

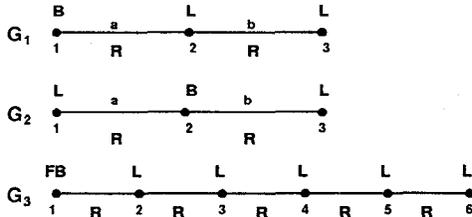


Fig 9. Specialized graphs for Ex. 5.1 and 5.2

EXAMPLE 5.1: Suppose we construct a 3-link hybrid robot from module sets: $\text{LINK} = \{B, L, L\}$ and $\text{JOINT} = \{R, R\}$. The first step is to construct its non-isomorphic kinematic graphs. In this case, the only possible tree structure for 3 vertices is a serially connected tree. From step 3, we find two non-isomorphic module assignments. Their specialized trees are shown in Fig. 9 and denoted by G_1 and G_2 . Consider G_1 first. Link 1 is a cube with one R-joints; there is only one distinct assembly configuration under this condition. There are 12 distinct assemblies for link 2, a prism, with 2 labeled R-joints, and 2 assemblies for link 3 with 1 R-joint. Altogether we can generate at most $N_{G_1} = 1 \times 12 \times 2 = 24$ constructions from G_1 . Similar for G_2 , we can find at most $N_{G_2} = 2 \times 2 \times 2 = 8$ possible constructions. However, G_2 is symmetric about the center vertex. $\mathcal{H}(G_2)$ contains two element: the identity, $((1, 2, 3), (a, b))$, and $((3, 2, 1), (b, a))$. Using this automorphism, we further reduce

the non-isomorphic constructions of G_2 to 6. In total, there are $24 + 6 = 30$ non-isomorphic constructions of a 3-link hybrid tree robot as shown in Fig. 10.

If we do not pay attention to equivalent constructions and enumerate them in a brute force fashion, for G_1 , there are 6 states for link 1, $\frac{10!}{2!8!} = 45$ states for link 2, and 10 states for link 3. There will thus be $6 \times 45 \times 10 = 1800$ constructions! For G_2 , there are 10 states for link 1, 15 states for link 2, and 10 states for link 3. In total, there will be $10 \times 15 \times 10 = 1500$ constructions! There are 3300 constructions altogether. Thus, our method provides a significant improvement over brute force enumeration. ■

EXAMPLE 5.2: Suppose we want to construct a 6-link 5-DOF fixed-base robot from $\text{LINK} = \{FB, L, L, L, L, L\}$ and $\text{JOINT} = \{R, R, R, R, R\}$ with a given kinematic graph G_3 shown in Fig.9, where FB stands for a fixed base. We assume that there is only one way to attach a revolute joint to the base. Furthermore, we restrict that each joint must be located at each end of the prism, hence, there are only 7 such distinct assembly states for Link 2, 3, 4, and 5. Link 6 has 2 states. Totally, there are at most $N_{G_3} = 1 \times 7 \times 7 \times 7 \times 7 \times 2 = 4802$ distinct constructions. Since G_3 has no symmetry, the automorphism group of G_3 contains only the identity element. The actual number of distinct configurations achieves the upper bound N_{G_3} . If we neglect the isomorphic assembly states on each link, there are 25 states for Link 2, 3, 4, and 5. (Each of the two joints can be attached either one of the 5 ports at the end of a prism, so there are 25 states for each link.) There will be $1 \times 25^4 \times 5 = 1,953,125$ constructions! ■

5.3. Computational Complexity Issues

Let us briefly consider the computational complexity of this algorithm. The tree generation algorithm in Step 1 is constant time for a given number of vertices. To find the automorphism group of a graph in step 2 requires an exhaustive search on isomorphisms of the graphs. Backtrack is basically an exponential time search algorithm. In step 3, the time to compute distinct assignments on a graph G_i under $\mathcal{H}(G_i)$ is $O(K^2)$, where K is the number of distinct assignments, since we are using the OrbitEnumerate algorithm.

In step 4-(a), we perform at most $|\mathcal{H}(G)|$ checks for the automorphism group of a specialized graph \mathcal{G} . In step 4-(b), the time to generate distinct assemblies on every link v_i is $O(N_{v_i}^2)$, where N_{v_i} is the number of distinct assemblies. Since many links in a hybrid will be of the same module type (and therefore have the same distinct assemblies) these assemblies can be calculated in advance and stored in a look-up table to save computation time. It is unnecessary to compute distinct assemblies for links having identical joint patterns.

Step 4-(c) in the procedure gives an upper bound, $N_{\mathcal{G}}$, on the number of distinct configurations for a given specialized kinematic graph \mathcal{G} . $N_{\mathcal{G}}$ equals the product of the number of distinct assemblies on every link in \mathcal{G} . Owing to symmetries of a specialized graph, the actual distinct configurations is always less than or equal to $N_{\mathcal{G}}$. The upperbound will be achieved only if the graph has no symmetry, i.e., the automorphism group defining the graph symmetry contains only the identity element. The sum of the $N_{\mathcal{G}}$'s for all the non-isomorphic graphs gives the upper bound on the number of n -link tree-like modular robot configurations.

In step 4-(d), we have to check the automorphisms of robot assembly configurations generated by previous step pairwise if the graph exhibits symmetry. Since step 4-(c) generates $N_{\mathcal{G}}$ eAIMs, we have to check $\binom{N_{\mathcal{G}}}{2}$ pairs of eAIMs. Generally speaking, $O(N_{\mathcal{G}}^2)$ checks are needed for one specialized graph \mathcal{G} .

Note that computationally costly graph isomorphism checks on labeled and specialized graphs are unnecessary in the enumeration process because we generate non-isomorphic graphs in the beginning of the procedure. There is no known polynomial time algorithm to check graph isomorphisms [10]. The eAIMs generated by step 4-(c) for one tree \mathcal{G} are structurally equivalent automatically. However, when two eAIMs are given without any previous knowledge of the underlying kinematic graphs, an isomorphism test is needed.

This discussion points out several features of the algorithm. First, its computational complexity depends on the properties of the link symmetry groups and the class of tree-like structures one is considering. It is thus difficult to give a precise bound on the computational complexity of this algorithm. Second, the computations are structured so as to avoid computationally expensive steps, such as graph isomorphism checking. Third, the reasonable computational complexity of the algorithm (and the examples of Section 5.2) implies that for almost any conceivable application, it is much more efficient to enumerate the non-isomorphic geometries with this algorithm, rather than using a brute force enumeration process.

6. Summary

This paper demonstrated a method to enumerate non-isomorphic assembly configurations of a tree-like homogeneous or hybrid modular robot from a set of specified modules. Non-isomorphic modular robots have distinct kinematic properties, and hence, different functionalities. We used kinematic graphs to represent candidate assembly configurations. We introduced a novel class of Assembly Incidence Matrices (AIMs) to represent the appropriate construction information. We also introduced a novel equivalence relationship on the AIMs based on the symmetric rotation groups of the links and the symmetries of the kinematic graph topology. These representations and equivalences form the basis for our algorithm. For demonstration purposes, we considered only a relatively simple set of link and joint modules. However, the method is completely general and can be applied to nearly any modular robotic system. Examples illustrated how this approach greatly improves the efficiency of enumeration process as compared to brute force enumeration. We are also investigating extensions of this method to enumerate non-equivalent robots for other types of equivalence relationships besides kinematic isomorphism.

Acknowledgments: This work was supported by the Caltech President's fund and NSF Grant MSS-9157843. We would also like to thank Dr. Guillermo Rodriguez for motivating this work.

References

- [1] E. Beckenbach, *Applied Combinatorial Mathematics*. New York, NY, John Wiley & Sons, 1964.
- [2] T. Beyer and S. M. Hedetniemi, "Constant Time Generation of Rooted Trees," *SIAM J. Computing*, 9, no. 4, pp. 706-712, 1980.
- [3] R. Cohen, M. G. Lipton, M. Q. Dai and B. Benhabib, "Conceptual Design of a Modular Robot," *ASME J. Mechanical Design*, 114, pp. 117-125, March, 1992.
- [4] N. Deo, *Graph theory with Applications to Engineering and Computer Science*. Englewood Cliffs, NJ, Prentice-Hall, 1974.
- [5] L. Dobrjanskyj and F. Freudenstein, "Some Applications of Graph Theory to the Structural Analysis of Mechanisms," *ASME J. Engineering for Industry*, 89, pp. 153-158, Feb., 1967.

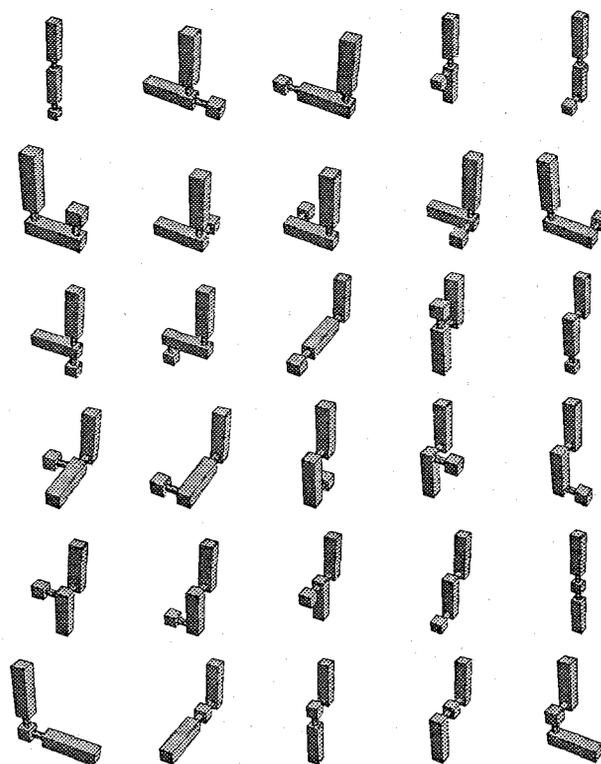


Fig 10. Distinct constructions of a 3-link hybrid robot

- [6] T. Fukuda and S. Nakagawa, "Dynamically Reconfigurable Robotic System," in *Proc. IEEE Int. Conf. Robotics and Auto.* pp. 1581-1586, 1988.
- [7] I. N. Herstein, *Topics in Algebra 2ed*. New York, NY, John Wiley & Sons, 1975.
- [8] C. L. Liu, *Introduction to Combinatorial Mathematics*. New York, NY, McGraw-Hill, 1968.
- [9] D. Schmitz, P. Khosla and T. Kanade, "The CMU Reconfigurable Modular Manipulator System," Carnegie Mellon Univ., CMU-RI-TR-88-7, 1988.
- [10] S. Skiena, *Implementing Discrete Mathematics*. Redwood City, CA, Addison-Wesley, 1990.
- [11] D. Stewart, R. Volpe and P. Khosla, "Integration of Real-Time Software Control Modules for Reconfigurable Sensor-based Systems," in *Proc. IEEE/RSJ Int. Workshop Intell. Robots and Systems*. 1992.
- [12] S. Williamson, *Combinatorics for Computer Science*. Rockville, MD, Computer Science Press, 1985.
- [13] L. S. Woo, "Type Synthesis of Plane Linkages," *ASME J. Engineering for Industry*, 89, pp. 159-172, Feb., 1967.
- [14] R. A. Wright, B. Richmond, A. Odlyzko and B. D. McKay, "Constant Time Generation of Free Trees," *SIAM J. Computing*, 15, no. 2, pp. 540-548, 1986.
- [15] K. H. Wurst, "The conception and Construction of a Modular Robot System," in *Proc. 16th Int. Sym. Industrial Robotics (ISIR)*. pp. 37-44, 1986.
- [16] H. -S. Yan and Y. -W. Hwang, "The specialization of Mechanisms," *Mechanism and Machine Theory*, 26, no. 6, pp. 541-551, 1991.