Conformational Switching in Self-Assembling Mechanical Systems

Kazuhiro Saitou, *Member, IEEE*

***Abstract*— A study of one-dimensional (1-D) self-assembly of a type of mechanical conformational switches, minus devices is presented where assembly occurs via the sequential mating of a random pair of parts selected from a part bin, referred to as sequential random bin-picking. Parametric design optimization of the minus devices via a genetic algorithm maximizing the yield of a desired assembly, and rate equation analyzes of the resulting designs, reveal that the minus devices facilitate the robust yield of a desired assembly against the variation in the initial fraction of the part types, by specifying a fixed assembly sequence during the self-assembling process. It is also found that while the minus devices can "encode" some assembly sequences, encoding other assembly sequences requires the use of another type of conformational switches, plus devices. To investigate the "encoding power" of these conformational switches, a formal model of self-assembling systems, one-dimensional self-assembling automaton, is introduced, where assembly instructions are written as local rules that specify conformational changes realized by the conformational switches. It is proven that the local rules corresponding to the minus and plus devices, and three conformations per each component, can encode any assembly sequences of a one-dimensional assembly of distinct components with arbitrary length.**

***Index Terms*— Assembly grammars, assembly sequences, genetic algorithms, mechanical conformational switches, self-assembling mechanical systems.**

I. Introduction

MANY complex biological structures arise from a process of self-assembly—assembly via random interactions among components. Biologists believe that assembly instructions for such self-assembly are built in each component molecules in the form of *conformational switches,* a mechanism that changes component shape as a result of local interactions among other components. In a protein molecule with several bond sites, for instance, a conformational switch causes the formation of a bond at one site to change the conformation of another bond site. As a result, a conformational change occurred at an assembly step provides the essential substrate for assembly at the next step, realizing a fixed assembly sequence self-assembly [2].

Designing self-assembling mechanical systems with such built-in assembly instructions is of great interest from an engineering point of view. If assembly of a mechanical system can occur via a bulk random agitation, e.g., shaking, it would be ideal for assembly of very small parts where the conventional robotic pick-and-place style assembly is extremely difficult or time-consuming. Also, since assembly instructions are distributed among each part rather than concentrated to an assembly robot, the assembly is more robust against unpredictable disturbances to the systems, such as malfunctions of the assembly robot. This would make the systems suitable to automated assembly in unstructured/unmodeled environment. As in their biological counterparts, conformational switches would also play an important role in such self-assembling mechanical systems.

The goal of this work is the fundamental understanding of the role of conformational switches in self-assembling mechanical systems. Section III discusses a case study of one-dimensional self-assembly of a type of mechanical conformational switches, *minus devices,* via the sequential mating of a random pair of parts selected from a part bin. In the case study, it is found that while the minus devices can "encode" some assembly sequences by introducing precedent constraints to the system, encoding other assembly sequences requires the use of another type of conformational switches, *plus devices*. To investigate the "encoding power" of these conformational switches, Section IV introduces a formal model of self-assembling systems, one-dimensional *self-assembling automaton,* where assembly instructions are written as local rules that abstract conformational changes realized by the conformational switches. Before proceeding to these results, Section II briefly discusses some related work on self-assembling mechanical systems.

II. Related Work

Several attempts have been made toward self-assembly of small mechanical parts to avoid direct part grasping. Moncevicz and Jakiela [3] developed layered palletization technique using Sony's automated parts orienting system (APOS), which "palletizes" a type of parts on top of another, by using vibration to convey them over a plastic pallet. Yeh and Smith [4] integrated trapezoidal gallium arsenide (GaAs) micro blocks on a silicon (Si) substrate with trapezoidal holes by dispensing these in a carrier fluid (ethanol) onto the Si substrate. Hosokawa *et al.* [5] experimented with the self-assembly of magnetized micro parts which are brought together on a water surface by magnetic force and surface tension of the water. Böhringer *et al.* [6] demonstrated bulk manipulation of millimeter scale electrical components by the combination of electrostatic

Manuscript received January 5, 1998; revised March 8, 1999. This paper was recommended for publication by Associate Editor E. Pagello and Editor P. B. Luh upon evaluation of the reviewers' comments. This work was supported in part by NSF Grant DDM-9058415 and matchable funds from Schlumberger, Inc.

The author is with the Department of Mechanical Engineering and Applied Mechanics, University of Michigan, Ann Arbor, MI 48109-2125 USA (e-mail: kazu@umich.edu).

Publisher Item Identifier S 1042-296X(99)04976-9.

filed and ultrasonic vibration. The most notable is the recent work by Terfort and Whitesides [7] which demonstrated self-assembly of a millimeter-scale light-emitting diode (LED) with an operating electrical circuit in aqueous environment. Two components for an LED are self-assembled in water by the effect of the complementary mating surface on each component, and the hydrophobic material coated on the mating surface. While these works realize assembly without direct part manipulation, there are at most *two* types of parts involved in a self-assembly, due to the lack of mechanisms which enforce precedence relationship during the self-assembly, such as conformational switches.

Although not termed as mechanical conformational switches, a few works incorporated them in attempts to develop and analyze self-assembling mechanical systems. Penrose [8] suggested several designs of mechanical conformational switches that are used in devices that "self-reproduce." These conformational switches cause a bond at one location to break a bond existing at another location or prevent a bond from occurring at another location. When the correct number and arrangement of subdevices are linked, the conformational switches cause the entire chain to cleave into two copies of the original self-reproducing device in a process akin to cell division. Another example is found in Hosokawa *et al.* [9]. They developed triangular parts employing switches realized with movable magnets that allow parts to bond together to form hexagons. The switches allow a part to be either in an active or inactive state. An activated part can bond to an inactivated part, turning the part to an activated state. These parts are assembled in a rotating box randomizer. The amounts of each intermediate subassembly achieved agreed reasonably well with the predicted values obtained by techniques analogous to chemical kinetics. This work, however, neither addresses the optimization of conformational switch design to maximize yield, nor the assembly sequences encoded by conformational switches.

III. Case Study

This case study discusses the role of a type of mechanical conformational switches, *minus devices*, [10], in one-dimensional self-assembly via a sequential mating of a random pair of parts selected from a part bin called *sequential random bin-picking.* A part can form a "bond" with another part *only* at the left or right *bond sites* (hence forming a one-dimensional assembly) when the two mating bond sites have complimentary shapes. By changing the shape of one of the bond sites, the minus device allows the formation of a bond *only* after the conformational change.

A. Sequential Random Bin-Picking

In this case study, a simple model of one-dimensional self-assembly called *sequential random bin-picking* is employed. It is a process of sequential mating of a random pair of parts selected from a part bin, which initially contains a random assortment of parts [Fig. 1(a)]. Mating of a pair of parts is accomplished according to the following three steps, which are repeated until prespecified conditions are satisfied:

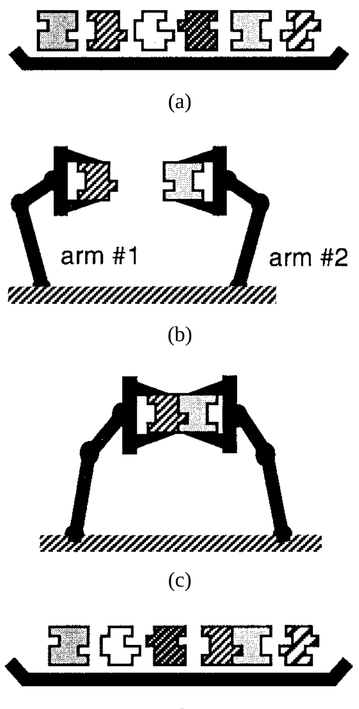

Fig. 1. Sequential random bin-picking.

Step 1: Arm #1 *randomly* picks up a part (possibly an assembly) from the bin. Then, arm #2 randomly picks up another part (possibly an assembly) from the bin [Fig. 1(b)].

Step 2: The two parts are pushed against each other, possibly forming an assembly [Fig. 1(c)].

Step 3: The parts are returned to the bin [Fig. 1(d)], possibly as an assembly.

It is assumed that a defect could occur with a certain probability at Step 2 when two chosen parts form an assembly.[1] If the defective assembly is chosen at Step 1 in subsequent iterations, no assembly can form at Step 2 *regardless of* the mating part, i.e., the parts in the defective assemblies are completely wasted.

Although having more than one pair of robot arms would model the explicit parallelism in assembly processes such as the ones in [5] and [7], the following examples assumed only one pair of robot arms to investigate the "worst case" scenario as an initial attempt. Note, however, that the above process can still simulate the simultaneous formation of multiple subassemblies, since it is globally synchronized by pairwise interactions, rather than by subassembly stages requiring multiple pairwise interactions.

B. Minus Devices

Fig. 2(a) shows a part with a minus device [10]. It consists of right and left bars that can slide horizontally, and one inner sliding block that can slide vertically. The left and right pictures in Fig. 2(a) show the minus device *before* and *after* conformational change, respectively. Before conformational

[1] If they cannot form an assembly, no defect occurs and they are simply returned to the bin.

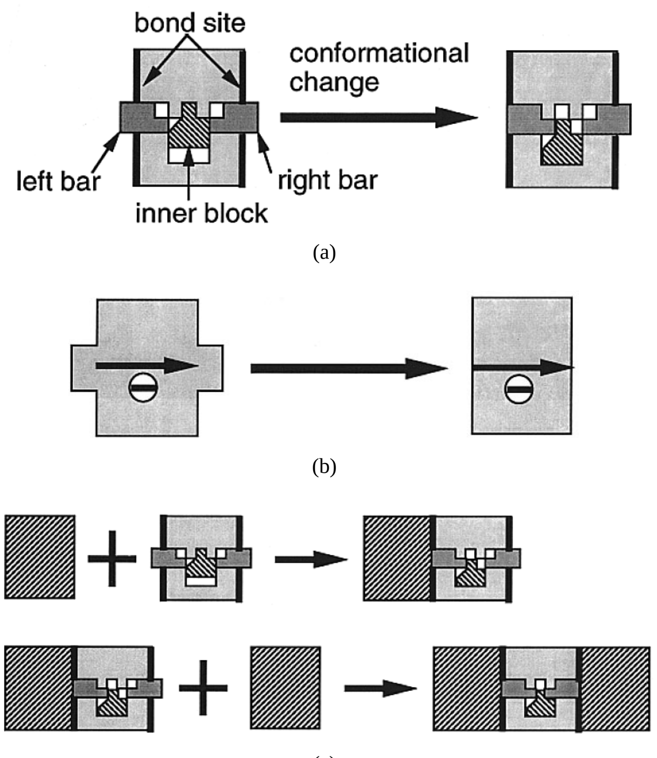

Fig. 2. Part with a minus device: (a) mechanism, (b) simplified notation, and (c) interaction with another part.

change, the left bar is free to be pushed in, whereas the right bar *cannot* be pushed in due to the position of the inner block. Conformational change of the device can be induced by pushing in the left sliding bar, which causes the inner block to slide down, which in turn creates a space for the right bar to slide in. As a result, the right bar can pushed in after conformational change. A simplified notation of the device is shown in Fig. 2(b), where an arrow indicates the direction in which one of the two horizontal bars can be pushed in to induce the conformational change. Note that in the simplified notation, the right sliding bar after conformational change is drawn as the "pushed-in" state, representing that the bar is "free" to be pushed in.

Fig. 2(c) illustrates how conformational change of the minus device can introduce a precedence constraint in assembly. Since only the left bar can be pushed in before the conformational change, a hatched rectangular part must come from the left in order to form a bond with the part with a minus device [top figure of Fig. 2(c)]. Then, it is only after the conformational change when another hatched rectangular part can form a bond from the right [bottom figure of Fig. 2(c)]. Since the right bar cannot be pushed in before the conformational change, first part *must* come from the left, introducing a left-to-right precedence constraint in assembly. Since the minus device is not "spring-loaded," it is impossible to reverse the conformational change. This implies that once a part changes its conformation and a bond is formed as a result, it will *never* be destroyed.

Note that a similar type of precedence constraints introduced by the conformational switching of the minus devices

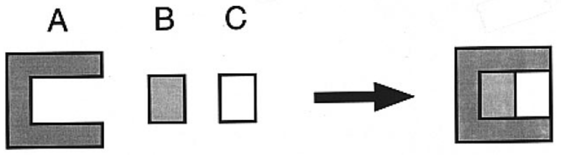

Fig. 3. Precedence constraints in rigid part assembly.

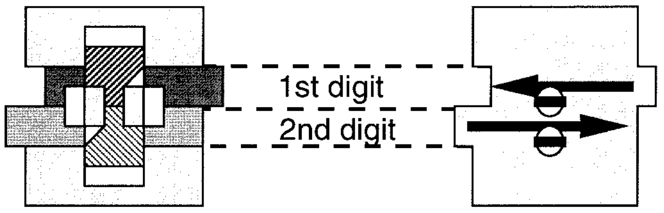

Fig. 4. Part with "two-digit" bonding sites.

appears in conventional rigid part assembly in two or three dimensions with certain mating surfaces. Fig. 3 illustrates such an example, where the geometrical relationship among A, B, and C implicitly imposes the constraint "A and B should be assembled before C." This type of precedence constraints have been extensively studied in assembly sequence planning in rigid part assembly [11]–[14]. This constraint, however, requires a certain geometrical relationship among A, B, and C, which may not be justifiable in designing parts that self-assemble. Also, without an explicit mechanism (e.g., a conformational switch) which *completely* rejects wrong a mating step, there are fairly large chances of a mating which violates a given precedence constraint in the process of self-assembly, where parts interact randomly one another. Conformational switching, on the other hand, can provide a reliable means of introducing precedence constraints in self-assembly without imposing any restrictions in geometrical relationship among parts.

The part design presented below has "two-digit" bonding sites, as shown in Fig. 4. The two-digit bonding sites are introduced in order to increase the number of possible shapes they can take, necessitated by the need of increasing the number of distinguishable part types. The shape of bond sites is represented by a pair of integers (a_1, a_2) referred to as *bond configuration.* Each component of a bond configuration takes 1 if the corresponding "digit" of the bond site has convex shape, -1 if the digit is concave, and 0 if it is flat. When bond sites of two parts meet, they form either:

1) stable bond;
2) unstable bond;
3) no bond.

The occurrences of these cases depend on the shape of the two mating bond sites, or equivalently, the bond configurations of the sites. Let (a_1, a_2) and (b_1, b_2) be the bond configurations of two bond sites contacting each other. These sites form a stable bond if they are complementary to each other, i.e., $a_1 + b_1 \leq 0$ and $a_2 + b_2 \leq 0$, and form an unstable bond if they are fairly complementary, i.e., $a_1 + b_1 = 1$ and $a_2 + b_2 \leq 0$, or $a_1 + b_1 \leq 0$ and $a_2 + b_2 = 1$. If none of the above applies, the two bond sites are not considered as complementary and

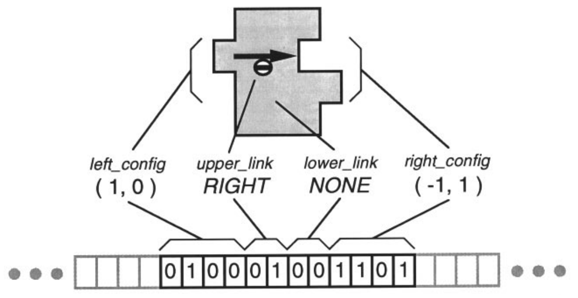

Fig. 5. Bit string representation of a part design.

therefore no bond is formed. An unstable bond potentially induces a conformational change of the involved bond sites. A conformational change can actually occur *only* when a stable bond is formed after the conformational change. Otherwise, neither a conformational change nor a bond is formed, and the two parts returned to the bin remain as separated.

C. Design Optimization of Minus Devices

In order to investigate the effect of the minus devices to the yield of a desired assembly during the process of sequential random bin-picking, parametric design optimization of the minus devices is performed. More precisely, the problem is stated as follows:

1) **Given:** The number of each type of parts in the bin, (i.e., the initial state of the bin), and defect probabilities.
2) **Find:** An optimal design of minus devices (and their initial bond configurations) that maximizes the yield of a desired assembly in the process of the sequential random bin-picking.

The above problem is formulated as search by parameterizing the design of the minus devices. Genetic algorithm [15], in conjunction with computer simulation of sequential random bin-picking, is used to search the parameter space of possible part designs.

The following four parameters uniquely specify a part two minus devices and two-digit bond sites: `left_config`, `right_config`, `upper_link`, and `lower_link`. `Left_config` and `right_config` are the initial bond configurations of the left bond site and the right bond site, respectively. `Upper_link` and `lower_link` are variables that specify the existence of the minus devices in a part, each of which takes one of the values `LEFT`, `RIGHT`, or `NONE`, depending on the direction of the arrow in the simplified notation of the corresponding minus devices. If `upper_link` is `LEFT`, for instance, there is a minus device at the "first digits" of the bond sites, with the arrow pointing to the left bond site. If `upper_link` is `NONE`, there is no minus device that connects the first digits of the bond sites, so they cannot undergo any conformational changes. Similarly, `lower_link` specifies the existence of the minus device between the second digits of the bond sites.

To apply genetic algorithm, the values of these parameters for *all types of* parts are represented as one binary string. As illustrated in Fig. 5, each part type has two bits for each component of `left_config` and `right_config`, and two bits for each of `upper_link` and `lower_link`. Since each part type requires 12 b, n types of part requires a bit string with the length $12n$.

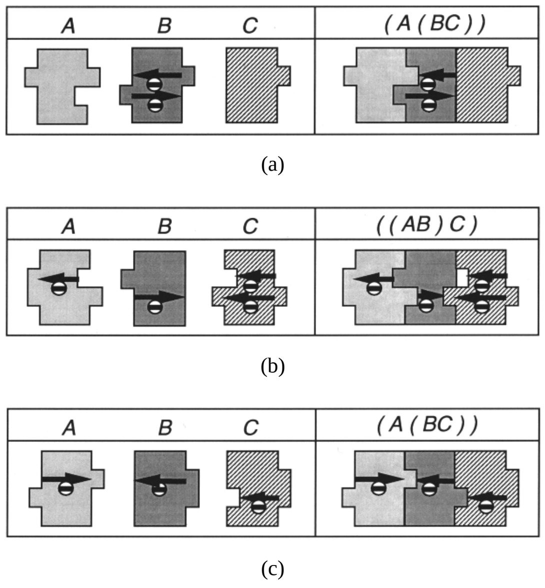

Fig. 6. Best part designs with $\mathbf{n}_0 = (10, 20, 10)$.

D. Self-Assembly with Three Part Types

First, the design optimization of the minus device as described above is done in the case of $n = 3$. The initial bin contains a random assortment of *three* types of parts, part A, part B, and part C, and the desired assembly whose yield is to be maximized is assembly ABC. The yield of ABC is computed as an average of the number of ABC's in the bin after 700 iterations of the sequential robot bin-picking, over 50 of such runs. The genetic algorithm in this example uses a crowding population [15] with 10% replacement per generation, fitness proportionate (roulette wheel) selection, and linear fitness scaling with scaling coefficient = 2.0. Also, the crossover probability is 0.9, the mutation probability is 0.03, the population size is 300, and the number of generations is 200. In the following results, $\mathbf{n}_0 = (n_A(0), n_B(0), n_C(0))$ is the vector of the initial numbers of parts A, B, and C in the bin, and $\mathbf{q} = (q_{AB}, q_{BC})$ is the vector of defect probabilities of the bonds between AB and BC, respectively.[2]

Fig. 6 shows the best part designs obtained in the case of $n_0 = (10, 20, 10)$, with three different defect probabilities: (a) $\mathbf{q} = (0.0, 0.0)$, (b) $\mathbf{q} = (0.2, 0.0)$, and (c) $\mathbf{q} = (0.0, 0.2)$. For $\mathbf{q} = (0.0, 0.0)$ and $\mathbf{q} = (0.0, 0.2)$, a part A can bind to a part B *only after* part B changes its conformation, which is triggered by the binding of part C. The formation of an assembly ABC, therefore, takes place through the two-step "reactions," $B+C \rightarrow B'C$ and $A+B'C \rightarrow AB'C$, where B' represents a part B after conformational change. Since no other reactions are possible, an ABC assembles *only* in the fixed assembly sequence $(A(BC))$. On the other hand, $((AB)C)$ is "encoded" in the best design with $\mathbf{q} = (0.2, 0.0)$. In this case, a part C can bind to a part B *only after* the part B changes

[2]It is assumed that the defect probability of a bond depends only on the parts associated to the bond, in particular, $q_{AB} = q_{A(BC)}$ and $q_{BC} = q_{(AB)C}$.

TABLE I
SUMMARY OF THE RESULTS: SELF-ASSEMBLY WITH THREE PART TYPES, A, B, AND C

q \ n0	(10,10,10)	(10,20,10)	(20,20,10)
(0.0,0.0)	{ ABC }	(A (BC))	((AB) C)
(0.2,0,0)	{ ABC }	((AB) C)	((AB) C)
(0.0,0.2)	{ ABC }	(A (BC))	(A (BC))

its conformation, which is triggered by binding of a part A as in $A + B \rightarrow AB'$ and $AB' + C \rightarrow AB'C$. Note that only one conformational link is actually used during the assembly of ABC in all three cases.

The total of nine optimization runs are done for three different cases of $\mathbf{n}_0$ and $\mathbf{q}$. The summary of these nine runs is shown in Table I, where $\{ABC\}$ represents no fixed assembly is specified by the minus devices, i.e., parts can be assembled in either $((AB)C)$ or $(A(BC))$. These results indicate the best part design specifies a fixed assembly sequence or no fixed assembly sequences, depending on the values of $\mathbf{n}_0$ and $\mathbf{q}$. It should be noted that genetic algorithm searches the space of possible part designs, *not* the space of assembly sequences. There is no explicit representation of assembly sequences in the above formulation. Rather, assembly sequences in Table I *emerged* due to a particular design of parts (with minus devices) that maximizes the yield of ABC.

E. Rate Equation Analyzes

Although the above results suggest the importance of assembly sequences for maximizing the yield of a desired assembly, it does not provide direct means of comparing the dynamic change of the part counts for different assembly sequences. This can be done by a discrete-time version of rate equation [9] for describing the kinetics of chemical systems [16]

$$\mathbf{n}(t+1) = \mathbf{n}(t) + \mathbf{A}\mathbf{p}(t) \tag{1}$$

where $\mathbf{n}(t)$ is a vector of the *expected* numbers of each possible assembly (including defected ones) at iteration t, $\mathbf{p}(t)$ is a vector of probabilities for each possible reaction at iteration t, and $\mathbf{A}$ is a matrix of stoichiometric coefficients of all possible reactions.

For each assembly sequences $((AB)C)$, $(A(BC))$, and $\{ABC\}$, (1) is numerically solved for 1000 iterations, which corresponds to 1000 iterations in the sequential robot bin-picking, with the nine conditions shown in Table I. The expected numbers of ABC, when assembled in the sequence $((AB)C)$, $(A(BC))$, and $\{ABC\}$, are then plotted in a plane to allow comparison among the three assembly sequences.

Fig. 7 is the resulting plots in the case of $\mathbf{n}_0 = (10, 20, 10)$, and $\mathbf{q} = (0.2, 0.0)$. The most noticeable fact in this figure, also observed in other cases with $\mathbf{n}_0 = (10, 20, 10)$, is the *very* low yield of $\{ABC\}$ compared to $((AB)C)$ and $(A(BC))$. This is due to the large number of the middle part B in $\mathbf{n}_0$, which produces a large number of AB's and BC's in the early stage of iterations if no fixed assembly sequence is specified. This excess production of AB's and BC's then causes the shortage of individual C's and A's later on, which are necessary to complete the final assembly ABC from the assemblies AB and BC, resulting in the low yield. By enforcing a fixed assembly sequence, $((AB)C)$ or $(A(BC))$, this excess production of AB and BC can be avoided.

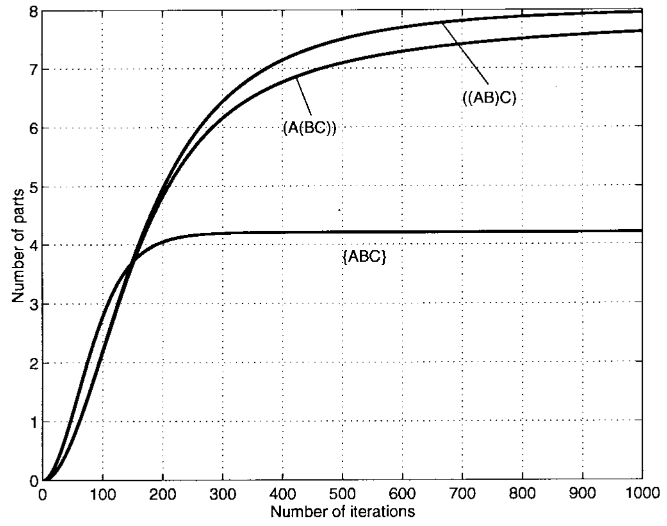

Fig. 7. Yield of ABC with $\mathbf{n}_0 = (10, 20, 10)$ and $\mathbf{q} = (0.2, 0.0)$.

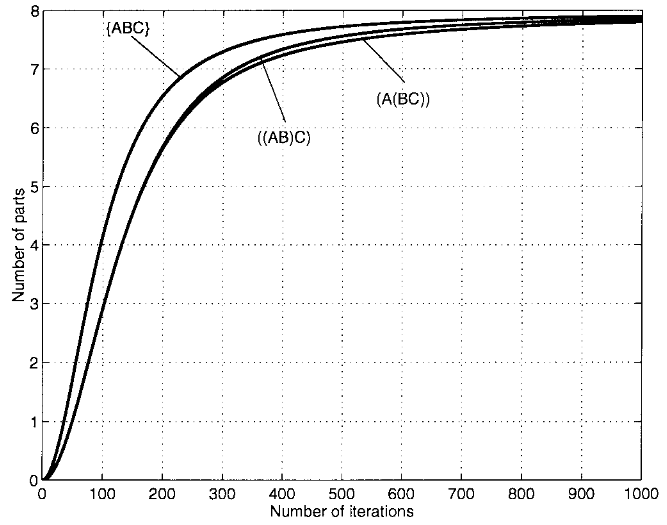

Fig. 8. Yield of ABC with $\mathbf{n}_0 = (10, 10, 10)$ and $\mathbf{q} = (0.2, 0.0)$.

Fig. 8 is the resulting plots in the case of $\mathbf{n}_0 = (10, 10, 10)$, and $\mathbf{q} = (0.2, 0.0)$. In this case, and also in the other cases with $\mathbf{n}_0 = (10, 10, 10)$, $\{ABC\}$ has the best yield, although the yield of the other two assembly sequences is almost as good. Since the fraction of the part types in the initial bin is equal to the fraction in the desired assembly, there is no chance of "wasting" subassembly as discussed above, and therefore, having no fixed assembly sequence, realized fastest production of the final assembly ABC. With the change in the initial fraction of part types, $\{ABC\}$ is outperformed by one of two fixed assembly sequences, as illustrated in Fig. 9 [the case with $\mathbf{n}_0 = (20, 20, 10)$ and $\mathbf{q} = (0.2, 0.0)$], and also in Fig. 7.

The above results suggest an important role of minus devices: facilitating the *robust* yield of a desired assembly against the variation in the initial fraction of the part types,

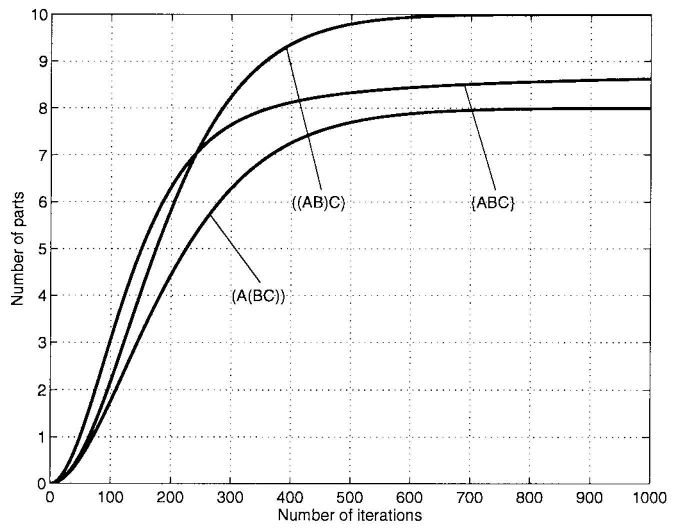

Fig. 9. Yield of ABC with $\mathbf{n}_0 = (20, 20, 10)$ and $\mathbf{q} = (0.2, 0.0)$.

by specifying a fixed assembly sequence during the self-assembling process. As stated in the previous section, the results illustrate the "worst case" scenario in terms of the speed of assembly—the number of iterations needed for convergence would dramatically reduce by the introduction of explicit parallelism to the assembly process, for instance, with multiple pairs of robot arms.

F. Self-Assembly with Four Part Types

The design optimization of the minus device is also done in the case of $n = 4$. The initial bin contains a random assortment of four types of parts, part A, part B, part C, and part D, and the desired assembly whose yield is to be maximized is assembly $ABCD$. The yield of $ABCD$ is computed as an average of the number of $ABCD$'s in the bin after 1400 iterations of the sequential robot bin-picking, over 50 of such runs. The genetic algorithm runs described in this section have a population size of 600, and the number of generations is 900. In the following results, $\mathbf{n}_0 = (n_A(0), n_B(0), n_C(0), n_D(0))$ is the vector of the initial numbers of parts A, B, C, and D in the bin, and $\mathbf{q} = (q_{AB}, q_{BC}, q_{CD})$ is the vector of defect probabilities of the bonds between AB, BC and CD, respectively. Note with $n = 4$, five fixed assembly sequences are possible: $(((AB)C)D)$, $((AB)(CD))$, $((A(BC)D))$, $(A((BC)D))$, and $(A(B(CD)))$.

The total of twelve optimization runs are done for three different cases of $\mathbf{n}_0$ and four different cases of $\mathbf{q}$. The summary of these twelve runs is shown in Table II, where $\{(AB)CD\}$ represents either $(((AB)C)D)$ or $((AB)(CD))$, and $\overline{\{(AB)CD\}}$ represents $((A(BC))D)$, $(A((BC)D))$ or $(A(B(CD)))$. Rate equations (1) are formulated in a similar way to the case of $n = 3$. The yield of the final assembly $ABCD$ is then compared for *all* assembly sequences which can be encoded by two minus devices, which are listed in Fig. 10. Among the eight assembly sequences listed, there are five ambiguous (partially fixed) assembly sequences, $\{ABCD\}$, $\{(AB)CD\}$, $\overline{\{(AB)CD\}}$, $\{AB(CD)\}$, and $\overline{\{AB(CD)\}}$, and three fixed assembly sequences, $(((AB)C)D)$, $((AB)(CD))$, and $(A(B(CD)))$. Note that the minus device *cannot* encode two of the fixed assembly sequences, $((A(BC))D)$ and $(A((BC)D))$. As in the case of $n = 3$, the results of design optimization in Table II show the emergence of various optimal assembly sequences depending on $\mathbf{n}_0$ and $\mathbf{q}$. This, and the associated rate equation analyzes (not shown due to the space limit) confirms the role of minus devices observed in the case of $n = 3$—facilitating the *robust* yield of a desired assembly against the variation in the initial fraction of the part types, by specifying a fixed assembly sequence during the self-assembling process. As in the case of $n = 3$, the results illustrate the "worst case" scenario in terms of the speed of assembly—the number of iterations needed for convergence would dramatically reduce by the introduction of explicit parallelism to the assembly process, for instance, with multiple pairs of robot arms.

TABLE II
SUMMARY OF THE RESULTS: SELF-ASSEMBLY WITH FOUR PART TYPES A, B, C, AND D

q \ n0	(10,10,10,10)	(10,20,10,10)	(10,20,20,10)
(0.0,0.0,0.0)	{ ABCD }	{ (AB) CD }	((AB) (CD))
(0.2,0,0,0.0)	{ ABCD }	{ (AB) CD }	((AB) (CD))
(0.0,0.2,0.0)	{ ABCD }	$\overline{\text{{ (AB) CD }}}$	(A(B(CD)))
(0.2,0.0,0.2)	((AB) (CD))	((AB) (CD))	((AB) (CD))

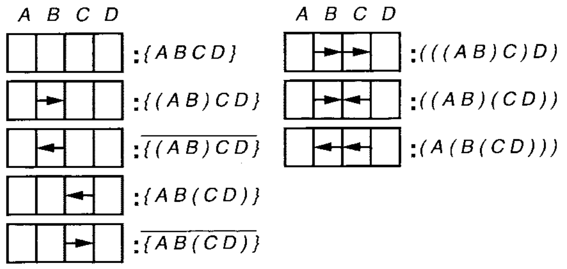

Fig. 10. Eight possible assembly sequences.

G. Encoding Power of Minus Devices

An important difference between the cases of $n = 3$ and $n = 4$, as noted earlier, is the existence of "unencodable" assembly sequences. In particular, one can quickly notice that the two assembly sequences $((A(BC))D)$ and $(A((BC)D))$ are unencodable with *no matter how* many minus devices being employed. In other words, these assembly sequences are beyond the encoding power of the minus devices. This suggests there could be an unencodable assembly sequence which yields better than any encodable assembly sequences. As shown in Fig. 11, in fact, the two unencodable assembly sequences $((A(BC))D)$ and $(A((BC)D))$ yield better than $(A(B(CD)))$, the best sequence obtained by the genetic algorithm with $\mathbf{n}_0 = (10, 20, 20, 10)$ and $\mathbf{q} = (0.0, 0.2, 0.0)$. It should be emphasized, however, that the sequence found by the genetic algorithm is the best among the assembly

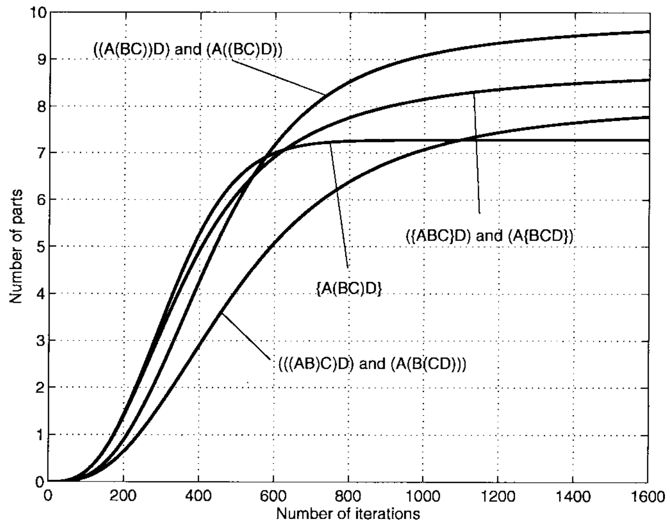

Fig. 11. Yield of $ABCD$ with $\mathbf{n}_0 = (10, 20, 20, 10)$ and $\mathbf{q} = (0.0, 0.2, 0.0)$: comparison with unencodable assembly sequences.

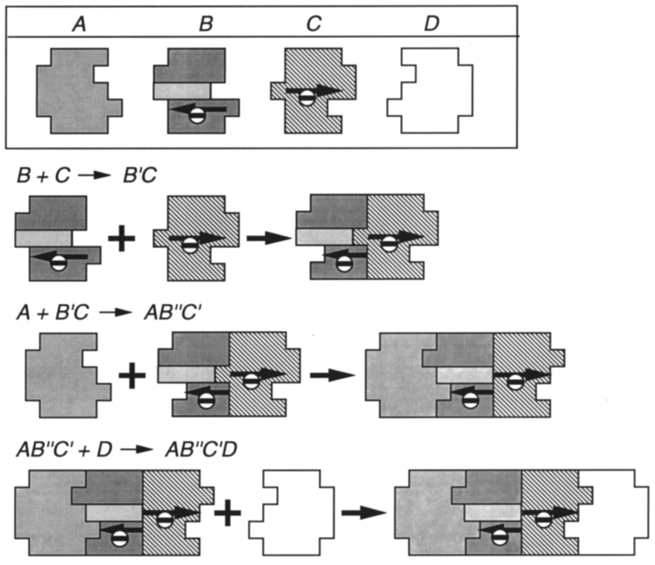

Fig. 12. Parts with conformational switches that encode $((A(BC))D)$.

sequences encodable to the minus devices. For comparison, Fig. 11 also shows the plot for three other unencodable assembly sequences, $(\{ABC\}D)$, $(A\{BCD\})$ and $\{A(BC)D\}$. The sequences $(\{ABC\}D)$ and $(A\{BCD\})$ also outperform the best encodable sequences $(((AB)C)D)$ and $(A(B(CD)))$.

In order to encode the sequences $((A(BC))D)$, it is necessary to introduce another type of conformational switch, a *plus* device, a simple sliding bar which propagates conformational change *through* multiple parts, and an additional "digit." Fig. 12 illustrates such parts with conformational switches that encode $((A(BC))D)$. Note that the part B has both a plus device and a minus device, and the part B has *three* conformations B, B', and B'', such that the *second* conformational change from B' to B'' upon the attachment of A is propagated to C through B. Other unencodable assembly sequences shown in Fig. 11 can also be encoded with the help of this plus device.

IV. Formal Model

In the previous section, it was found that minus devices can encode fixed or partially-fixed assembly sequences, which facilitates the robust yield of a desired assembly against the variation in the initial fraction of part types. It was also found that some assembly sequences are *unencodable* to minus devices no matter how many devices being employed, and such unencodable assembly sequences can be encoded by using *both* minus device *and* plus device, another type of conformational switches which propagates conformational changes through multiple components.

These findings raise the following two questions.

1) Is it possible to tell whether a given assembly sequence is encodable to minus devices, or to minus *and* plus devices?
2) If so, how many conformations (or switch states) are necessary to encode the given assembly sequence?

The relationship between assembly sequences and conformational switches is analogous to the one between languages and "machines" (models of computation) in the theory of computation [17], with an assembly sequence being an instance of a language, and a set of conformational switches that encodes the assembly sequence being a machine that accepts the instance of the language.

This analogy motivated us to study a formal model of self-assembling systems which abstracts the function of the minus and plus devices, and to identify classes of self-assembling systems based on the classes of assembly sequences in which the components of the systems self-assemble. Although conventional formal models, such as one-dimensional cellular automata [18] and automata networks [19] possess some similarity to the self-assembling systems discussed in Section III, their operation lacks direct analogy to self-assembly with conformational switching, where the components randomly interact one another and conformational changes occur in components as a result of local interactions with other components.

A new formal model, therefore, is developed which retains such direct analogy during its operation. The model, which we will refer to as an one-dimensional *self-assembling automaton*, is defined as a sequential rule-based machine that processes one-dimensional string of symbols (components), where the rules specify conformational changes of components realized by the minus and plus devices. Several theorems regarding the classes of self-assembling automaton are provided, although proofs are omitted due to page restrictions. Instead, the connection of each theorem to the case study in Section III is emphasized. The complete proofs to all theorems are found in [20] and [21]. Due to the abstract nature of the model, the presentation of this section is more formal than the previous ones.

A. Definition of Self-Assembling Automata

We abstract a component (part) to be an element of a finite set Σ, and an assembly to be a string in Σ^+. Additionally, a component $a \in \Sigma$ can take a finite number of conformations represented by $a, a', a'', \cdots$, and the transition among conformations is specified by a set of assembly rules, which abstracts the function of conformational switches. Each component, therefore, can be viewed as a finite automaton that self-assembles. Note that in the definition below, attaching rules are an abstraction of minus devices, and propagation rules are an abstraction of the plus devices.

Definition 1: A *one-dimensional self-assembling automaton* (SA) is a pair $M = (\Sigma, R)$, where Σ is a finite set of *components,* and R is a finite set of *assembly rules* of the form either $a^\alpha + b^\beta \rightarrow a^\gamma b^\delta$ *(attaching rule)* or $a^\alpha b^\beta \rightarrow a^\gamma b^\delta$ *(propagation rule),* where $a, b \in \Sigma$ and $\alpha, \beta, \gamma, \delta \in \{'\}^*$.[3] The *conformation set* of $a \in \Sigma$ is a set $Q_a = \{a^\alpha | \alpha \in \{'\}^*, a^\alpha$ appears in $R\}$. The *conformation set* of M is the union of all conformation sets of $a \in \Sigma$.

As in the case of the sequential robot bin-picking, we view an SA as having an associated *component bin,* with an infinite number of "slots," each of which can store an assembly or the null string Λ. Initially, a finite number of the slots contain assemblies and the rest of the slots are filled with Λ. Self-assembly of components proceeds by *selecting* a random pair of assemblies or an assembly in the component bin, and *applying* the rules in R to the selected assemblies. As a result of the rule application, assemblies are *deleted* from and *added* to the component bin, just like assertions are deleted from and added to working memory in rule-based inference systems. The rule application is done according to the following procedure, where $a, b \in \Sigma$ and $\alpha, \beta, \gamma, \delta \in \{'\}^*$.

1) If a pair of assemblies $(x, y) = (za^\alpha, b^\beta u)$ for some $z, u \in \Sigma^*$ is selected, and R contains the rule $r = a^\alpha + b^\beta \rightarrow a^\gamma b^\delta$ (r *fires*), *delete* x and y, and *add* $za^\gamma b^\delta u$.
2) If an assembly $x = za^\alpha b^\beta u$ for some $z, u \in \Sigma^*$, and R contains the rule $r = a^\alpha b^\beta \rightarrow a^\gamma b^\delta$ (r *fires*), *delete* x and *add* $za^\gamma b^\delta u$.

If neither of the above applies, the selected assemblies are simply returned to the component bin, leaving the bin unchanged. Note that the random pick and the rule application model Steps 1–3 of the sequential robot bin-picking. Note also that at any point of self-assembly, the component bin contains a finite number of nonnull strings with finite length, since the total number of components in the initial bin is finite and no new components are created when applying the rules to the bin.

We define $\texttt{SEQ}(A)$ as the language generated by the context-free grammar $\forall a \in A, S \rightarrow (SS)|a$. Note that $A \subset \texttt{SEQ}(A)$. A string x in $\texttt{SEQ}(A)$ is a full parenthesization of a string $u = \texttt{RM-PAREN}(x)$ in A^+, where RM-PAREN is a function that removes parentheses from its argument string. We interpret the parse tree of x as a (binary) assembly tree, i.e., a representation of a pairwise assembly sequence of u.

Definition 2: Let Σ be a component set of an SA. An *assembly sequence* is a string in $\texttt{SEQ}(\Sigma)$. An assembly sequence x is *basic* if x contains at most one copy of elements in Σ, i.e., $\forall a \in \Sigma, N_a(x) \leq 1$.

Definition 3: Let $M = (\Sigma, R)$ be an SA. A *configuration* of M is a bag $\langle x | x \in \texttt{SEQ}(Q)\rangle$, where Q is the conformation set of M. Let $x \in \texttt{SEQ}(\Sigma)$ be an assembly sequence. A configuration Γ *covers* x if $\Gamma = \langle a | a \in \Sigma\rangle$ and $\forall a \in \Sigma, N_a(x) \leq N_a(\Gamma)$.

[3]It is assumed $a^\Lambda = a$, where Λ is the null string.

The sequence of self-assembly can be traced by examining the configuration each time the component bin changes as a result of applying the rules in R to the component bin. To keep track of the order of assembly, the nonnull strings newly added to the component bin are parenthesized in the new configuration if they were added by an attaching rule.

For two configurations Γ and Φ, we write $\Gamma \vdash_M \Phi$ if the configuration of M changes from Γ to Φ as a result of applying a rule in R to the component bin *exactly once,* reading "Φ is *derived* from Γ *at one step.*" Similarly, $\Gamma \vdash_M^* \Phi$ if the configuration of M changes from Γ to Φ as a result of applying the rules in R to the component bin *zero or more times,* reading "Φ is *derived* from Γ." If there is no ambiguity, $\vdash_M$ and $\vdash_M^*$ are often shortened to $\vdash$ and $\vdash^*$, respectively.

Example 1: Consider a SA $M_1 = (\Sigma_1, R_1)$ where $\Sigma_1 = \{a, b, c\}$ and $R_1 = \{a + b \rightarrow ab', b' + c \rightarrow b'c\}$. Let $\Gamma = \langle a, b, c, c\rangle$ and $\Phi = \langle a, b, c\rangle$. The configurations Γ and Φ cover the assembly sequence $((ab)c)$. Self-assembly of $((ab)c)$ from Γ proceeds as $\langle a, b, c, c\rangle \vdash_{M_1} \langle (ab'), c, c\rangle \vdash_{M_1} \langle ((ab')c), c\rangle$.

Note that the SA in Example 1 is an abstraction of the three part self-assembly discussed in Section III-D.

Given an SA as defined above, the process of self-assembly eventually terminates when no rule firing is possible, or runs forever due to an infinite cycle of rule firing. It is natural to say an SA self-assembles a given string in a given sequence if the process of self-assembly terminates, and *all* terminating configurations contain the string that is assembled in the sequence. This is a conservative definition, requiring *stable* and *reliable* production of the string assembled in the sequence. Formally, this can be stated as follows:

Definition 4: Let $M = (\Sigma, R)$ be an SA, Γ be a configuration of M and $x \in \texttt{SEQ}(\Sigma)$ be an assembly sequence. Γ is *stable* if there is no rule firing from Γ, i.e., $C_M(\Gamma) = \{\Gamma\}$, where $C_M(\Gamma) = \{\Phi | \Gamma \vdash_M^* \Phi\}$. M *self-assembles* x from Γ if the both of the following hold:

1) All configurations derived from Γ can derive a stable configuration, i.e., $\forall \Phi \in C_M(\Gamma), \exists \Phi_1 \in C_M(\Phi), C_M(\Phi_1) = \{\Phi_1\}$.
2) $\forall \Phi \in C_M^*(\Gamma), \exists y \in \Phi$ such that $x = \texttt{RM-PRIME}(y)$, where $C_M^*(\Gamma)$ is a set of stable configurations derived from Γ, and RM-PRIME is a function that removes the prime symbols ($'$) from its argument.

Example 2: M_1 in Example 1 self-assembles $((ab)c)$ from $\langle a, b, c, c\rangle$.

B. Classes of Self-Assembling Automata

The two classes of SA are defined based on the presence of propagation rules in the rule set. Note that in the definition below, a class I SA corresponds to a set of parts with *only* minus devices, and a class II SA corresponds a set of parts with *both* plus *and* minus devices.

Definition 5: Let $M = (\Sigma, R)$ be an SA. M is *class I* if R contains *only* attaching rules. M is *class II* if R contains *both* attaching rules *and* propagation rules.

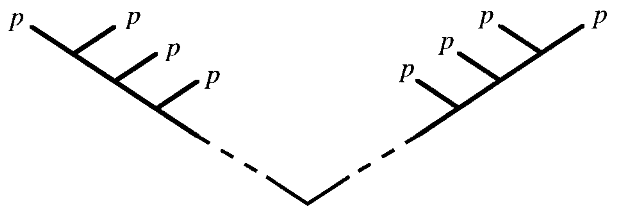

Fig. 13. Parse tree of an assembly template generated by G_I.

Definition 6: An *assembly template* is a string $t \in \mathtt{SEQ}(\{p\})$. An *instance* of t on a finite set Σ is an assembly sequence $x \in \mathtt{SEQ}(\Sigma)$ obtained by replacing p in t by $a \in \Sigma$. If x is an instance of t, t is an *assembly template of* x.

Example 3: Two strings $t_1 = ((pp)(pp))$ and $t_2 = ((p(pp))p)$ are assembly templates. Let $\Sigma = \{a, b, c, d\}$. The basic assembly sequences $x_1 = ((ab)(cd))$ and $x_2 = ((b(ad))c)$ are instances of t_1 and t_2 on Σ, respectively.

Definition 7: An *assembly grammar* is a context-free grammar with a language that is a subset of $\mathtt{SEQ}(\{p\})$. The class I assembly grammar G_I is defined by the following production rules:

$$\begin{aligned} S &\rightarrow (LR) \\ L &\rightarrow (Lp)|p \\ R &\rightarrow (pR)|p. \end{aligned}$$

The parse tree of the assembly templates in $L(G_I)$ is shown in Fig. 13. Each of the left and right subtrees is a *linear* assembly tree, which specifies self-assembly proceeding in one direction. The parse trees of the assembly templates in $\mathtt{SEQ}(\{p\})$ are general binary tree with no special structures.

We can interpret $L(G_I)$ and $\mathtt{SEQ}(\{p\})$ as sets of assembly templates with different numbers of changes in the direction of self-assembly. Let t be an assembly template and x be an instance of t. If $t \in L(G_I)$, the direction of self-assembly does *not* alter during the self-assembly of x. If $t \in \mathtt{SEQ}(\{p\})\backslash L(G_I)$, the direction of self-assembly alters *least once* during the self-assembly of x. Based on these observations, the following theorems can be proven. Here we again state only the facts.

Theorem 1: For any basic assembly sequence x that is an instance of an assembly template $t \in L(G_I)$, there exists a class I SA which self-assembles x from a configuration that covers x.

Theorem 2: For any basic assembly sequence x that is an instance of an assembly template $t \in \mathtt{SEQ}(\{p\})\backslash L(G_I)$, there exists a class II SA which self-assembles x from a configuration that covers x. Further, there exist *no* class I SA which self-assembles x from a configuration that covers x.

Note that the assembly sequence $(((AB)C)D)$, $((AB)(CD))$, and $(A(B(CD)))$ listed in Fig. 10 are instances of an assembly template $t \in L(G_I)$. Theorem 1 assures that these assembly sequences are indeed encodable with only minus devices. On the other hand, the assembly sequence $(A((BC)D))$ in Fig. 12 encodable with both plus and minus devices is an instance of an assembly template $t \in \mathtt{SEQ}(\{p\})\backslash L(G_I)$. Theorem 2 assures that this assembly sequence is indeed encodable with both plus and minus devices, but unencodable with minus devices only.

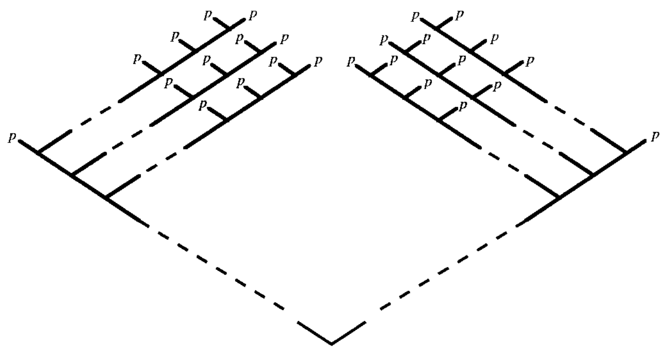

Fig. 14. Parse tree of an assembly template generated by G_{II}.

C. Minimum Conformation SA

In this section, we provide the minimum number of conformations necessary to encode a given assembly sequence based on the classes of basic assembly sequences introduced earlier. Since the number of conformations may vary for each component, the following definition is necessary.

Definition 8: Let M be an SA and Q is the conformation set of M. M is an SA with n conformations if $n = \max_{a^\alpha \in Q} |\alpha|$.

Definition 9: The class II assembly grammar G_{II} is defined by the following production rules:

$$\begin{aligned} S &\rightarrow (L_0 R_0) \\ L_0 &\rightarrow (L_0 R_1)|R_1 \\ R_0 &\rightarrow (L_1 R_0)|L_1 \\ L_1 &\rightarrow (L_1 p)|p \\ R_1 &\rightarrow (pR_1)|p. \end{aligned}$$

Note that $L(G_I) \subset L(G_{II}) \subset \mathtt{SEQ}(\{p\})$. The parse tree of the assembly templates in $L(G_{II})$ is shown in Fig. 14. The parse tree in Fig. 14 can be obtained from the parse tree in Fig. 13, by replacing leaves at the right branches of the left subtree by a linear assembly tree, and *vice versa.* Let x be an assembly sequence and t is an assembly template of x. If $t \in L(G_{II})\backslash L(G_I)$, the direction of self-assembly alters *exactly once,* and if $t \in \mathtt{SEQ}(\{p\})\backslash L(G_{II})$, the direction of self-assembly alters *more than once* during the self-assembly of x.

Example 4: The assembly template t_2 in Example 3 cannot be generated by G_I but can be generated by G_{II}, for example, through the derivation $S \Rightarrow (L_0 R_0) \Rightarrow ((L_0 R_1) R_0) \Rightarrow ((pR_1)R_0) \Rightarrow ((p(pR_1))R_0) \Rightarrow ((p(pp))R_0) \Rightarrow ((p(pp))p)$ and hence $t_2 \in L(G_{II})\backslash L(G_I)$. An assembly template $t_3 = (p((p(pp))p))$ cannot be generated by G_{II} and hence $t_3 \in \mathtt{SEQ}(\{p\})\backslash L(G_{II})$.

The minimum number of conformations of SA that are necessary to self-assemble a given basic assembly sequence x depends on whether x is an instance of an assembly template in $L(G_I)$, $L(G_{II})\backslash L(G_I)$, or $\mathtt{SEQ}(\{p\})\backslash L(G_{II})$.

Theorem 3: For any basic assembly sequence x that is an instance of an assembly template $t \in L(G_I)$, there exists a class I SA M with two conformations which self-assembles x from a configuration Γ that covers x. For $\mathtt{L}(x) \geq 3$, M is an SA with the minimum number of conformations which self-assembles x from Γ.

Theorem 4: For any basic assembly sequence x that is an instance of an assembly template $t \in L(G_{II})\backslash L(G_I)$, there exists a class II SA M with two conformations which self-assembles x from a configuration Γ that covers x. And M is an SA with the minimum number of conformations which self-assembles x from Γ.

Example 5: Consider $\Sigma = \{a, b, c, d\}$ and $x = ((a(bc))d)$. The assembly sequence x is an instance of $t_2 = ((p(pp))p)$ in Example 3. From Example 4, $t_2 \in L(G_{II})\backslash L(G_I)$. Let R contain the following rules: $b+c \rightarrow b'c$, $a+b' \rightarrow ab$, $bc \rightarrow bc'$, and $c'+d \rightarrow c'd$. It is clear that $M = (\Sigma, R)$ is an SA with two conformations which self-assembles x from the configurations that cover x, e.g., $\langle a, b, c, d\rangle$ and $\langle a, a, b, b, c, c, d, d\rangle$.

Note R in Example 5 corresponds to the part design in Fig. 12, which has three conformations for part B. Three conformations, instead of two as stated in Theorem 4, were needed, since a plus device and a minus device cannot be implemented in one "digit."

Next, we claim that only *three* conformations are necessary to encode an *arbitrary* x. This might sound counter-intuitive, since we are claiming that only three conformations can encode basic assembly sequences with arbitrary (possibly *very* large) sizes. The proof of the claim is based on the observation that there are only two kinds of propagation rules: the rules that propagate conformational changes to the left, and the rules that propagate conformational changes to the right, and that for any given two adjacent components, these two kinds of propagation rules *always* fire in alternate order. Complete proof of the claim, as stated below, is found in [20] and [21].

Theorem 5: For any basic assembly sequence x that is an instance of an assembly template $t \in \mathtt{SEQ}(\{p\})\backslash L(G_{II})$, there exists a class II SA M with three conformations which self-assembles x from a configuration Γ that covers x. And M is an SA with the minimum number of conformations which self-assembles x from Γ.

Example 6: Consider $\Sigma = \{a, b, c, d, e\}$ and $x = (a((b(cd))e))$. The assembly sequence x is an instance of $t_3 = (p((p(pp))p))$ in Example 4, and $t_3 \in \mathtt{SEQ}(\{p\})\backslash L(G_{II})$. Let R contain the following rules: $c+d \rightarrow c'd$, $b+c' \rightarrow bc''$, $c''d \rightarrow c''d'$, $d'+e \rightarrow d''e$, $c''d'' \rightarrow cd''$, $bc \rightarrow b'c$, and $a+b' \rightarrow ab'$. It is clear that $M = (\Sigma, R)$ is an SA with three conformations which self-assembles x from the configurations that cover x, e.g., $\langle a, b, c, d, e\rangle$ and $\langle a, b, b, c, d, d, e, e\rangle$.

Theorem 5 provides the theoretical lower bound to the number of conformations needed to encode arbitrary assembly sequences. It simply states *no* implementation of the conformational switches can encode arbitrary assembly sequences with less than three conformations per component. A particular physical implementation of the conformational switches, however, may or may not be able to encode arbitrary assembly sequences with this theoretical lower bound. For instance, R in Example 6 cannot be realized by using the plus and minus devices as they are now, due to the physical limitation in implementing these two types of switches in one "digit." It is possible, however, to construct an R that encodes arbitrary assembly sequences with n conformations per component, even if the conformational switches only allow "nonoverlapping digits." An algorithm to construct such an R and its correctness proof can be found in [21].

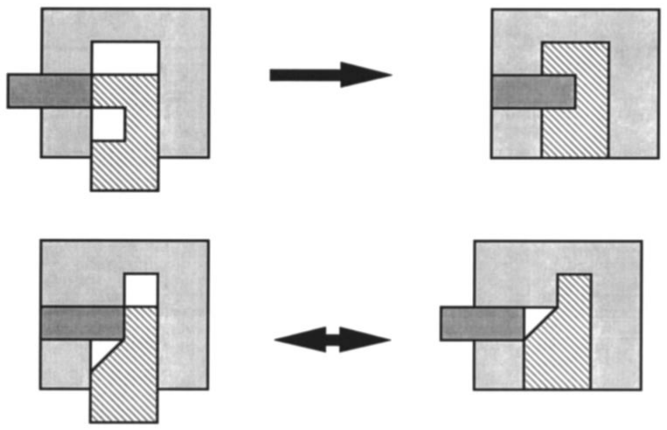
Fig. 15. Examples of 2-D conformational switch designs: equivalents of the minus device (top) and the plus device (bottom).

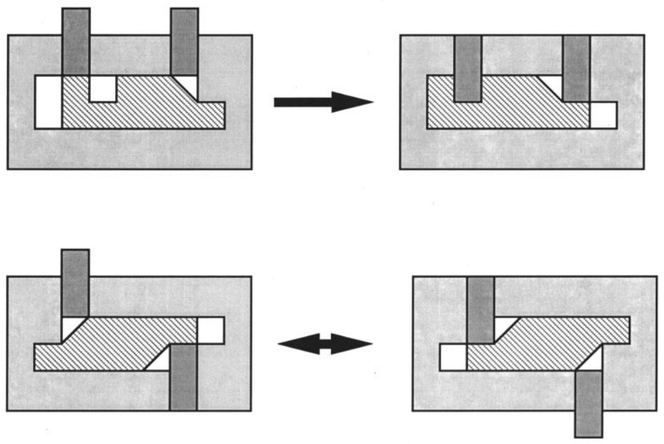
Fig. 16. Examples of 2-D conformational switch designs: equivalents of the minus device (top) and the plus device (bottom).

V. Conclusion

This paper discussed a case study of one-dimensional self-assembly of a type of mechanical conformational switches, *minus devices,* where assembly occurs via the sequential mating of a random pair of parts selected from a part bin, referred to as *sequential random bin-picking.* Parametric design optimization of the minus devices maximizing the yield of a desired assembly, and rate equation analyzes of the resulting designs, revealed that the minus devices facilitated the *robust* yield of a desired assembly against the variation in the initial fraction of the part types, by specifying a fixed assembly sequence during the self-assembling process. It was also found that while the minus devices could "encode" some assembly sequences, encoding other assembly sequences required the use of another type of conformational switches, *plus devices.*

To investigate the "encoding power" of these conformational switches, a formal model of self-assembling systems, one-dimensional *self-assembling automaton*, was introduced where assembly instructions were written as local rules that specified conformational changes of components realized by the minus and plus devices. Classes of self-assembling automata were defined based on three classes of assembly sequences described by assembly grammars. It was proven that the local rules corresponding to the minus and plus devices, and *three* conformations per each component, could encode *any* assembly sequences of a one-dimensional assembly of distinct components with *arbitrary* length.

The work presented in this paper was our first attempt toward the fundamental understanding on the role of conformational switching in self-assembling mechanical systems. Although based on quite simplistic models, the knowledge obtained in this work, as summarized above, will be beneficial to the design of self-assembling mechanical systems for practical applications. In particular, the theory of self-assembling automaton as discussed in this paper can also be applied to the self-assembly in 2- or 3-dimensions, since it is purely based on the assembly sequences in which the components self-assemble. Extensions are necessary, however, to the design of conformational switches in order to design self-assembling systems in higher dimensions. Figs. 15 and 16 illustrate examples of such extensions for two-dimensional (2-D) cases that might be incorporated in future applications.

We believe one of the most promising areas for application of self-assembly is the assembly of micro- to mesoscale components for microelectrical applications, where surface adhesion force causes extreme difficulty in direct part grasping and handling, [22]. As stated in Section II, several attempts have been already made to realize self-assembly of micro- to meso-scale electrical components, for example, using the gravitational field coupled with shape complementarity [4], using the electrostatic field coupled with ultrasonic agitation [6], and using the hydrophobic effect coupled with shape complementarity [7]. Applications of conformational switches to such self-assembling systems will dramatically increase the complexity of the system that can be self-assembled by the added capability to control precedence relationships among components during self-assembly. To do so, a simple mechanism and an energy supply to cause conformational change suitable to microelectrical application will be necessary. Design of mechanical conformational switches which overcome this problem is left for future development.[4]

Acknowledgment

The author would like to thank Dr. M. Jakiela for financial support and supervision during the early stage of this work, and the anonymous reviewers for their valuable comments.

[4]Depending on the application domains, however, the implementation of conformational switches does not need to be actually mechanical—Theorem 5 can be interpreted, for example, that a simple argumented finite state machine with $2 +, \log_2 n$ bits of memory (2 for conformations, and $\log_2 n$ for ID of component types) would be enough to encode arbitrary assembly sequences for n components!

References

[1] K. Saitou, "Conformational switching as assembly instructions in self-assembling mechanical systems," in *Proceedings of the International Conference on Complex Systems*, Y. Bar-Yam, Ed. Nashua, NH: Perseus, 1997. [Online]. Available: http://www.interjournal.org, 1998.

[2] J. D. Watson, N. H. Hopkins, J. W. Roberts, J. A. Steitz, and A. M. Weiner, *Molecular Biology of the Gene.* Menlo Park, CA: Benjamin/Cummings, 1987.

[3] P. H. Moncevicz, M. J. Jakiela, and K. T. Ulrich, "Orientation and insertion of randomly presented parts using vibratory agitation," in *Proceedings of the ASME 3rd Conference on Flexible Assembly Systems*, A. H. Soni, Ed. New York: Amer. Soc. Mech. Eng., Sept. 1991, vol. DE-33, pp. 41–47.

[4] H. J. Yeh and J. S. Smith, "Fluidic self-assembly of GaAs microstructures on Si substrates," *Sens. Mater.*, vol. 6, no. 6, pp. 319–332, 1994.

[5] K. Hosokawa, I. Shimoyama, and H. Miura, "Two-dimensional micro-self-assembly using the surface tension of water," in *IEEE J. Microelectromech. Syst.*, 1996.

[6] K. Böhringer, K. Goldberg, M. Cohn, R. Howe, and A. Pisano, "Parallel microassembly with electrostatic force fields," in *Proc. 1997 IEEE Int. Conf. Robot. Automat.*, 1997.

[7] A. Terfort and G. M. Whitesides, "Self-assembly of an operating electrical circuit based on shape complementarity and the hydrophibic effect," *Adv. Mater.*, vol. 10, no. 6, pp. 470–473, 1998.

[8] L. S. Penrose, "Self-reproducing machines," *Sci. Amer.*, vol. 200, pp. 105–114, Jun. 1959.

[9] K. Hosokawa, I. Shimoyama, and H. Miura, "Dynamics of self-assembling systems: Analogy with chemical kinetics," *Artif. Life*, vol. 1, no. 4, pp. 413–427, 1994.

[10] K. Saitou and M. J. Jakiela, "Subassembly generation via mechanical conformational switches," *Artif. Life*, vol. 2, no. 4, pp. 377–416, 1995.

[11] A. Bourjault, "Contribution a une Approche Méthodologique de L'Assemblage Automatisé: Elaboration Automatique des Séquences Opératoires," Ph.D. thesis, Univ. Franche-Comté, Besançon, France, Nov. 1984.

[12] T. de Fazio and D. Whitney, "Simplified generation of all mechanical assembly sequences," *IEEE J. Robot. Automat.*, vol. RA-3, pp. 640–658, Dec. 1987; corrections vol. RA-4, pp. 705–708, Dec. 1988.

[13] L. Hommem de Mello and A. Sanderson, "A correct and complete algorithm for the generation of mechanical assembly sequences," *IEEE Trans. Robot. Automat.*, vol. 7, pp. 228–240, Apr. 1991.

[14] S. Lee and Y. Shin, "Assembly planning based on geometric reasoning," *Comput. Graph.*, vol. 14, no. 2, pp. 237–250, 1990.

[15] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[16] J. I. Steinfeld, J. S. Francisco, and W. L. Hase, *Chemical Kinetics and Dynamics.* Englewood Cliffs, NJ: Prentice-Hall, 1989.

[17] J. C. Martin, *Introduction to Language and the Theory of Computation.* New York, NY: McGraw-Hill, 1991.

[18] A. R. Smith III, "Simple computation-universal cellular spaces," *J. Assoc. Comput. Mach.*, vol. 18, no. 3, pp. 339–353, July 1971.

[19] K. Culik II and J. Karhumäki, "On totalistic systolic networks," *Inform. Process. Lett.*, vol. 26, no. 5, pp. 231–236, 1988.

[20] K. Saitou, "Conformational switching in self-assembling mechanical systems: Theory and application," Ph.D. dissertation, Mass. Inst. Technol., Cambridge, Jun. 1996.

[21] K. Saitou and M. J. Jakiela, "On classes of one-dimensional self-assembling automata," *Complex Systems*, vol. 10, no. 6, pp. 391–416, Dec. 1996.

[22] R. S. Fearing, "Survey of sticking effects for micro parts handling," in *IEEE J. Microelectromech. Syst.*, pp. 212–217, 1995.

Kazuhiro Saitou (M'97) received the Ph.D. degree in mechanical engineering from the Massachusetts Institute of Technology, Cambridge, in 1996.

He is an Assistant Professor, Department of Mechanical Engineering and Applied Mechanics, University of Michigan, Ann Arbor. His research interests include design optimization, design for assembly, micro assembly, and evolutionary computation in design.